Introduction about Jamboree

Jamboree is one-stop solution for students who want to study abroad. It is basically divided into three heads: Training, Admissions, and Visa. The institite makes entire process of studying abroad (from preparing to joining foreign university), smooth process. It has introduced services such as discounted couriers, educational loans and pre-departure orientation etc. It recently launched a feature which estimates the chances of graduate admission from an Indian perspective. In this feature, students can check their probability of getting into the IVY league college

Problem Statement

Analyze what factors are important in graduate admissions and how these factors are interrelated among themselves?

The probability of getting into the IVY league college depends on several factors such as:

1. Scores obtained in associated exam
2. Academic History
3. Effectiveness of statement of purpose or cover letter.
4. Strength of Letter of Recommendation
5. Presence of any other outstanding performance other than academic
6. Rating or demand of University for which student has applied.

Approach for solution

1. Need to analyze how much weightage these features hold to get admission and hence conclude the importance of features.
2. Need to check if these factors are related or not.
3. Need to check if one factor affect others.
4. Need to build model

Exploratory Data Analysis

In [108…

```
# import all libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [109…

```
import warnings
warnings.filterwarnings('ignore')
```

In [110…

```
#import the dataset
dataset = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/(
dataset.head()
```

Out[110]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [111…

```
# Make copy of dataset
df = dataset.copy()
```

In [112…

```
#Structure or shape of DATASET
df.shape

## The dataset has 500 records or entries with 9 features
```

Out[112]:

```
(500, 9)
```

In [113…

```
#check datatypes of all attributes
df.info()

## Datatype of features like Serial No., GRE Score, TOEFL Score, University Rating
## Datatype of features like SOP,LOR,CGPA and Chance of Admit is of float type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

In [114…

```
# Check missing Value in dataset
df.isnull().sum()

## There are no null values in dataset
```

Out[114]:

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [115…

```
# Check Statistical summary of dataset
```

```
# Before checking statistical summary of dataset, it can be observed that the first
# just the unique ids. There is no need of analyzing statistics summary of this fea

df.drop(["Serial No."],axis=1,inplace=True)
df.head(2)
```

Out[115]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76

In [116…

```
# Now Check Statistical summary of dataset
df.describe()
```

Out[116]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chan of Adn
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.721
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.141
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.340
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.630
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.720
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.820
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.970

Statistical Summary of all features of dataset

1. There are total 500 entries for all features.

GRE Score

1. The mean and standard deviation of GRE Score are 316.47 and 11.295 respectively.
2. The mininum and maximum value of GRE Score in dataset are 290 and 340 respectively.
3. The 25th, 50th and 75th percentile for GRE score are approx 308,317 and 325 respectively.

It can be observed that the variables are continuous for this feature

TOEFL Score

1. The mean and standard deviation of TOEFL Score are 107.2 and 6.08 respectively.
2. The mininum and maximum value of TOEFL Score in dataset are 92 and 120 respectively.
3. The 25th, 50th and 75th percentile for TOEFL score are approx 103,107 and 112 respectively.

It can be observed that the variables are continuous for this feature

University Rating

1. The mean and standard deviation of University Rating are 3.11 and 1.14 respectively.
2. The mininum and maximum value of University Rating in dataset are 1 and 5 respectively.
3. The 25th, 50th and 75th percentile for University Rating are approx 2,3 and 4 respectively.

It can be observed that the variables are discrete for this feature. The possible values are 1,2,3,4 and 5.

Statement of purpose (SOP) Feature

1. The mean and standard deviation of SOP are 3.37 and 0.99 respectively.
2. The mininum and maximum value of SOP in dataset are 1 and 5 respectively.
3. The 25th, 50th and 75th percentile for SOP are approx 2.5,3.5 and 4 respectively.

It can be observed that the variables are discrete for this feature. The possible values are 1,2,3,4 and 5.

Letter of Recommendation (LOR) Feature

1. The mean and standard deviation of LOR are 3.48 and 0.92 respectively.
2. The mininum and maximum value of LOR in dataset are 1 and 5 respectively.
3. The 25th, 50th and 75th percentile for LOR are approx 3,3.5 and 4 respectively.

It can be observed that the variables are discrete for this feature. The possible values are 1,2,3,4 and 5.

CGPA

1. The mean and standard deviation of CGPA are 8.57 and 0.60 respectively.
2. The mininum and maximum value of CGPA in dataset are 6.8 and 9.92 respectively.
3. The 25th, 50th and 75th percentile for CGPA are approx 8.13,8.56 and 9.04 respectively.

It can be observed that the variables are continuous for this feature

Research

1. The mean and standard deviation of Research are 0.56 and 0.49 respectively.
2. The mininum and maximum value of Research in dataset are 0 and 1 respectively.
3. The 25th, 50th and 75th percentile for Research are approx 0,1 and 1 respectively.

By observing above values of this feature, it can be said they have values either 0 or 1.

Chance of Admit

1. The mean and standard deviation of Chance of Admit are 0.72 and 0.14 respectively.
2. The mininum and maximum value of Chance of Admit in dataset are 0.34 and 0.97 respectively.
3. The 25th, 50th and 75th percentile for Chance of Admit are approx 0.63,0.72 and 0.82 respectively.

Non-Graphical Analysis of all Features

In [117…

```
from IPython.display import display, HTML
CSS = """
.output {
 flex-direction: row;
}
"""
HTML('<style>{}</style>'.format(CSS))
```

Out[117]:

In [118…

```
df_cols_cont = ['GRE Score','TOEFL Score','CGPA','Chance of Admit ']
for i in df_cols_cont:
    d1 = pd.DataFrame(df[i].value_counts().reset_index())
    d1.columns = [i,"Count"]
    display(d1)

# It can be confirmed that features 'GRE Score', 'TOEFL Score' and 'CGPA' holds co
```

	GRE Score	Count
0	312	24
1	324	23
2	316	18
3	321	17
4	322	17
5	327	17
6	311	16
7	320	16
8	314	16
9	317	15
10	325	15
11	315	13
12	308	13
13	323	13
14	326	12
15	319	12
16	313	12
17	304	12
18	300	12
19	318	12
20	305	11
21	301	11
22	310	11
23	307	10
24	329	10
25	299	10
26	298	10
27	331	9
28	340	9
29	328	9
30	309	9
31	334	8
32	332	8
33	330	8
34	306	7
35	302	7

	GRE Score	Count
36	297	6
37	296	5
38	295	5
39	336	5
40	303	5
41	338	4
42	335	4
43	333	4
44	339	3
45	337	2
46	290	2
47	294	2
48	293	1

	TOEFL Score	Count
0	110	44
1	105	37
2	104	29
3	107	28
4	106	28
5	112	28
6	103	25
7	100	24
8	102	24
9	99	23
10	101	20
11	111	20
12	108	19
13	113	19
14	109	19
15	114	18
16	116	16
17	115	11
18	118	10
19	98	10
20	119	10
21	120	9
22	117	8
23	97	7
24	96	6
25	95	3
26	93	2
27	94	2
28	92	1

	CGPA	Count
0	8.76	9
1	8.00	9
2	8.12	7
3	8.45	7
4	8.54	7
...	...	...
179	9.92	1
180	9.35	1
181	8.71	1
182	9.32	1
183	7.69	1

184 rows × 2 columns

	Chance of Admit	Count
0	0.71	23
1	0.64	19
2	0.73	18
3	0.72	16
4	0.79	16
...	...	...
56	0.38	2
57	0.36	2
58	0.43	1
59	0.39	1
60	0.37	1

61 rows × 2 columns

In [119…

```
df_cols_disc = ['University Rating','SOP', 'LOR ', 'Research']
for i in df_cols_disc:
    d1 = pd.DataFrame(df[i].value_counts().reset_index())
    d1.columns = [i,"Count"]
    display(d1)

# It can be observed that features 'University Rating', 'SOP' and 'LOR' holds discr
# 'University Rating' ranges from 1 to 5 whereas 'SOP' and 'LOR' features ranges fr
# Feature 'Research' holds binary values either 0 or 1
```

	University Rating	Count
0	3	162
1	2	126
2	4	105
3	5	73
4	1	34

	SOP	Count
0	4.0	89
1	3.5	88
2	3.0	80
3	2.5	64
4	4.5	63
5	2.0	43
6	5.0	42
7	1.5	25
8	1.0	6

	LOR	Count
0	3.0	99
1	4.0	94
2	3.5	86
3	4.5	63
4	2.5	50
5	5.0	50
6	2.0	46
7	1.5	11
8	1.0	1

	Research	Count
0	1	280
1	0	220

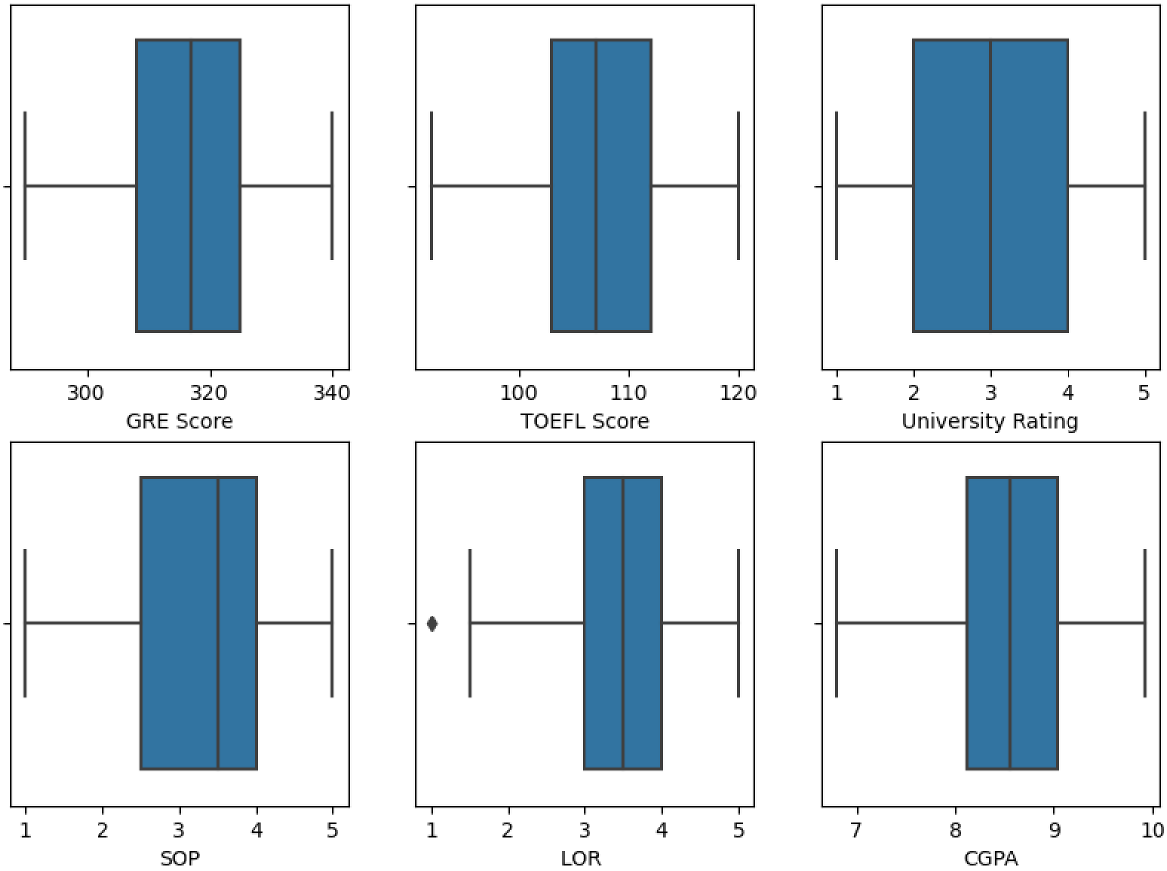
Outliers

Visual Analysis of Outliers present in dataset*

In [120…

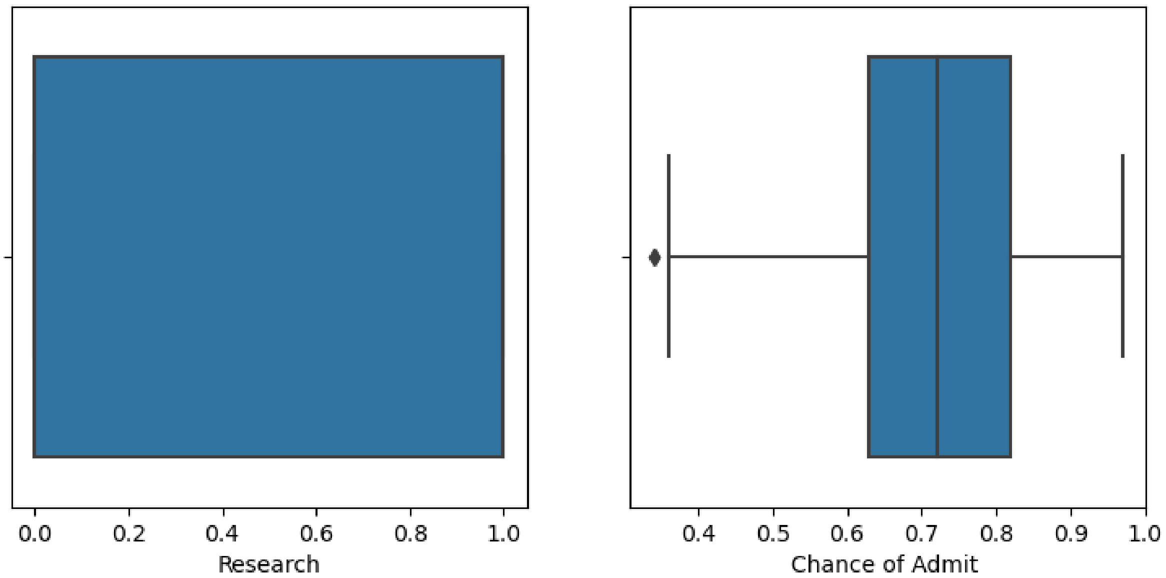
```
#plotting box plots to detect outliers in the data
df_cols = ['GRE Score','TOEFL Score','University Rating','SOP','LOR ','CGPA']
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(10, 7))
index = 0
```

```
for row in range(2):
    for col in range(3):
        sns.boxplot(x=df[df_cols[index]], ax=axis[row, col])
        index += 1
plt.show()
```



In [121…

```
fig, axs = plt.subplots(1, 2, figsize=(9,4))
sns.boxplot(x=df['Research'], ax=axs[0])
sns.boxplot(x=df['Chance of Admit '], ax=axs[1])
plt.show()
```



Observation on Outliers

1. There are no outliers in 'GRE Score','TOEFL Score','University Rating','SOP','CGPA' and 'Research' Features.

2. There is only one outlier in LOR and Chance of Admit feature.
3. The percentage of outlier present in feature 'LOR' is ((1/500)multiplies(100)) equals 0.2%, which is too low.
4. Similarly, percentage of outlier present in 'Chance of Admit' is 0.4% which is very low.

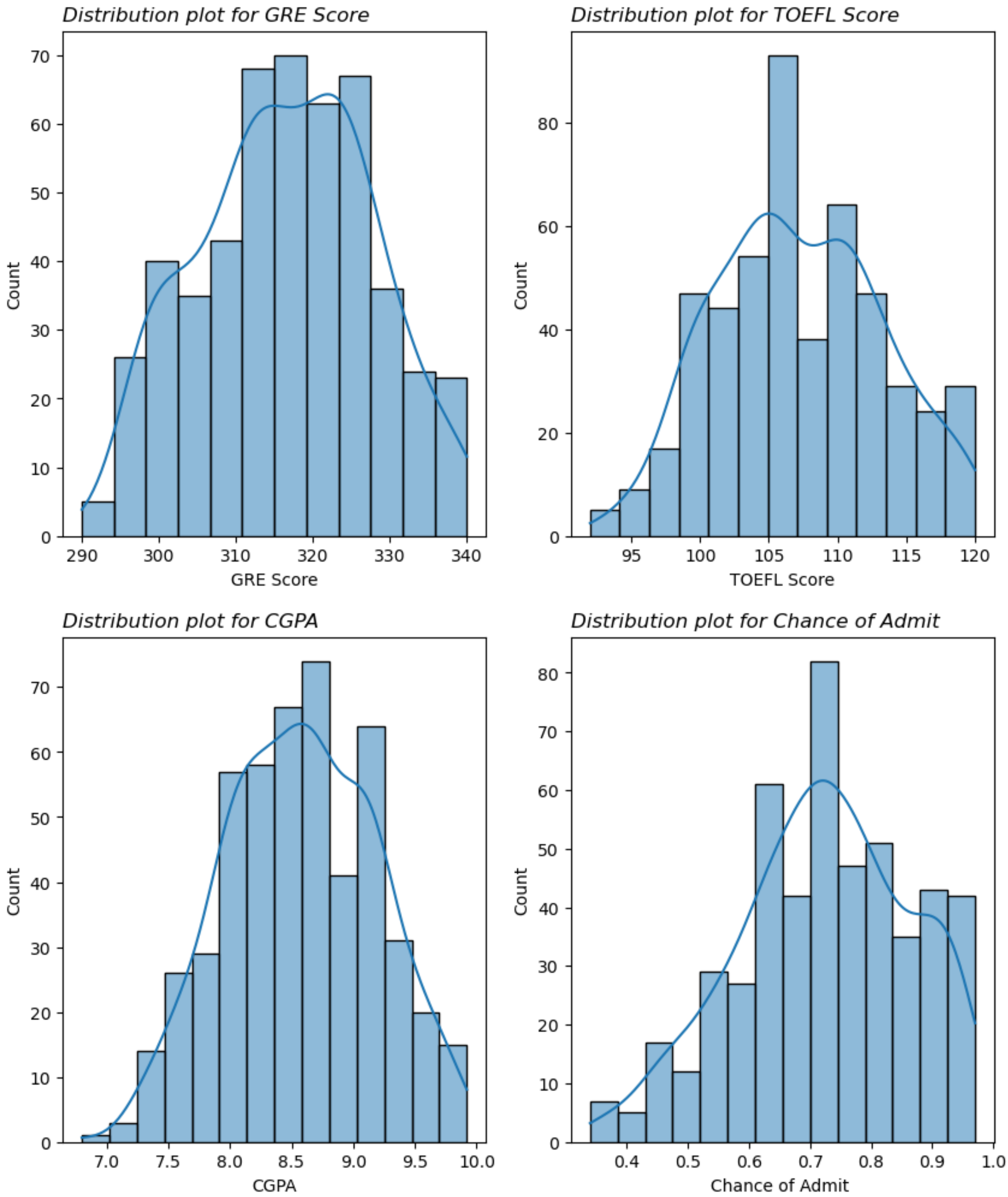
Need to identify the data point belongs to outlier (Refer part 2 of this report)

Univariate Analysis

Univariate analysis of features have continuous values

In [122…

```
#Univariate analysis of features have continuous values
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10, 12))
index = 0
for row in range(2):
    for col in range(2):
        sns.histplot(df[df_cols_cont[index]], ax=axis[row, col], kde=True).set_titl
                                +df_cols_cont[index],loc="left",fontstyle='italic'

        index += 1
plt.show()
```



Distribution plot for GRE Score

1. The range of GRE score obtained is from 290 to 340.
2. As it can be observed, majority marks obtained by students lying in range of 310 to 328.
3. Few students scored below 300 and approx 50 students scored above 330.

Distribution plot for TOEFL Score

1. The range of TOEFL score obtained is from 92 to 120.
2. As it can be observed,majority marks obtained by students lying in range of 103 to 107.
3. Few students scored below 100 and approx 30 students scored around 120.

Distribution plot for CGPA

1. The range of CGPA obtained is from 7 to 10.

2. As it can be observed, most CGPA points lying in range of 8 to 9.2.
3. A drastic increase in count of students can be observed from the students who scored below 8.

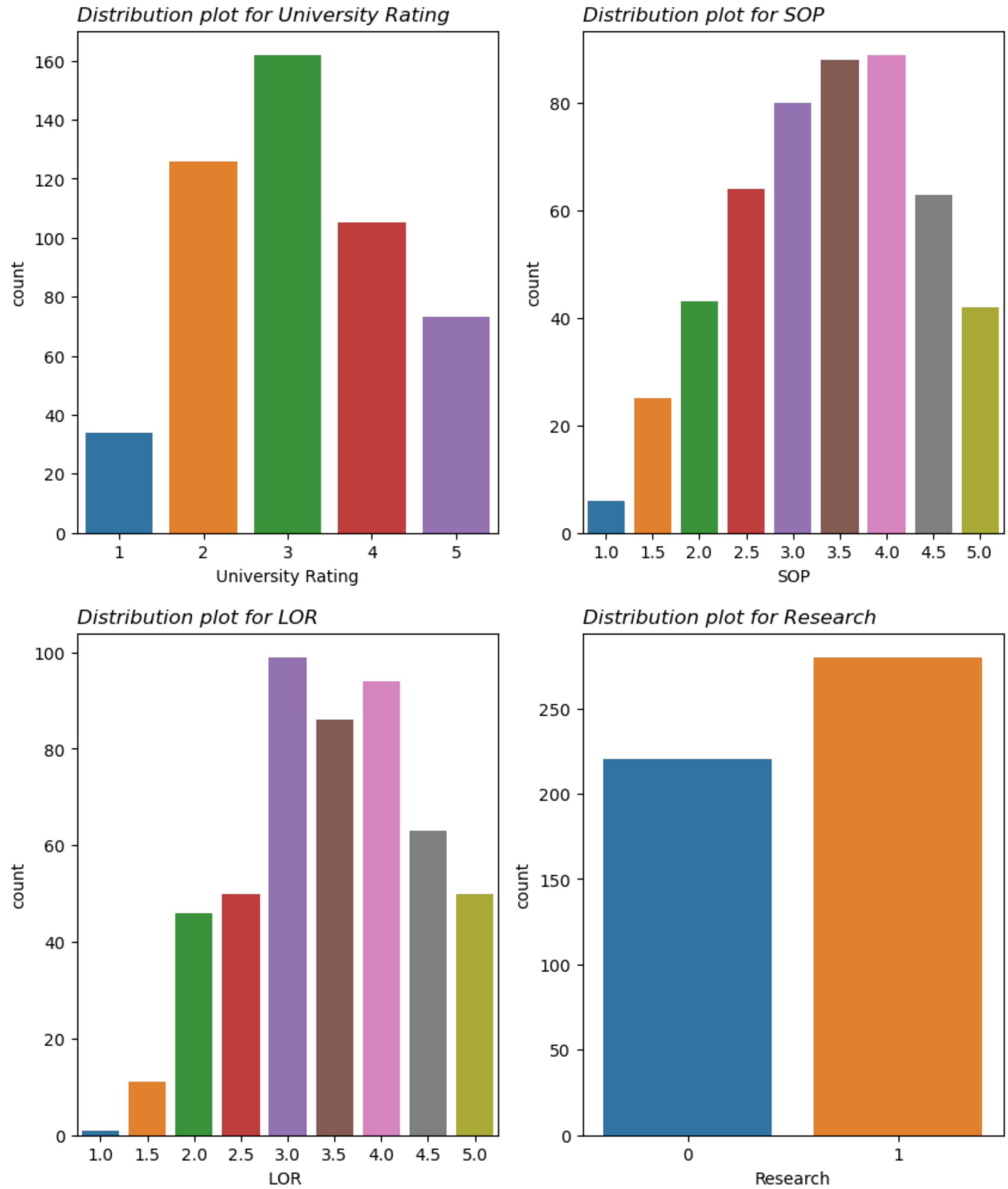
Distribution plot for Chance of Admission

1. The minimum possibility is around 40%.
2. Chance of admission for majority of students is around 70%.
3. Around 80-90 out of 500 students have channce of more than 85%.

Univariate analysis of features have discrete values

In [123…

```
#Univariate analysis of features have discrete and binary values
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10, 12))
index = 0
for row in range(2):
    for col in range(2):
        sns.countplot(df[df_cols_disc[index]], ax=axis[row, col]).set_title("Distr
                                     df_cols_disc[index],loc="left",fontstyle='italic')
        index += 1
plt.show()
```



Distribution plot for Rating of University

1. Most of the students prefer universities with rating as 3.
2. Around 70 studnets prefer university of rating 5.

Distribution plot for Statement of Purpose (SOP)

1. Most of the SOPs have been rated as 3.5 and 4 out of 5.
2. Very few around <10 SOP have bene rated 1
3. Around 50 out of 500 SOPs have been rated as 5.

Distribution plot for Letter of Recommendation (LOR)

1. Most of the LORs have been rated from 3 to 4 out of 5.
2. Very few around <10 SOP have bene rated 1.
3. Around 50 out of 500 LORs have been rated as 5.

Distribution plot for Research

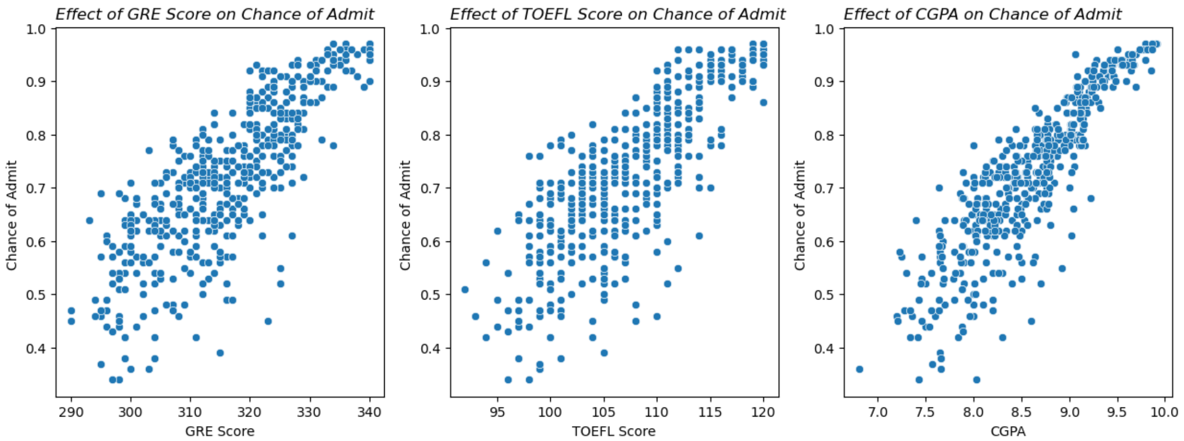
1. 280 out of 500 students have research.
2. 220 out of 500 studnets do not have reserach

Bivariate Analysis

Bivariate analysis of features having continuous values against the chance of admission

In [124...

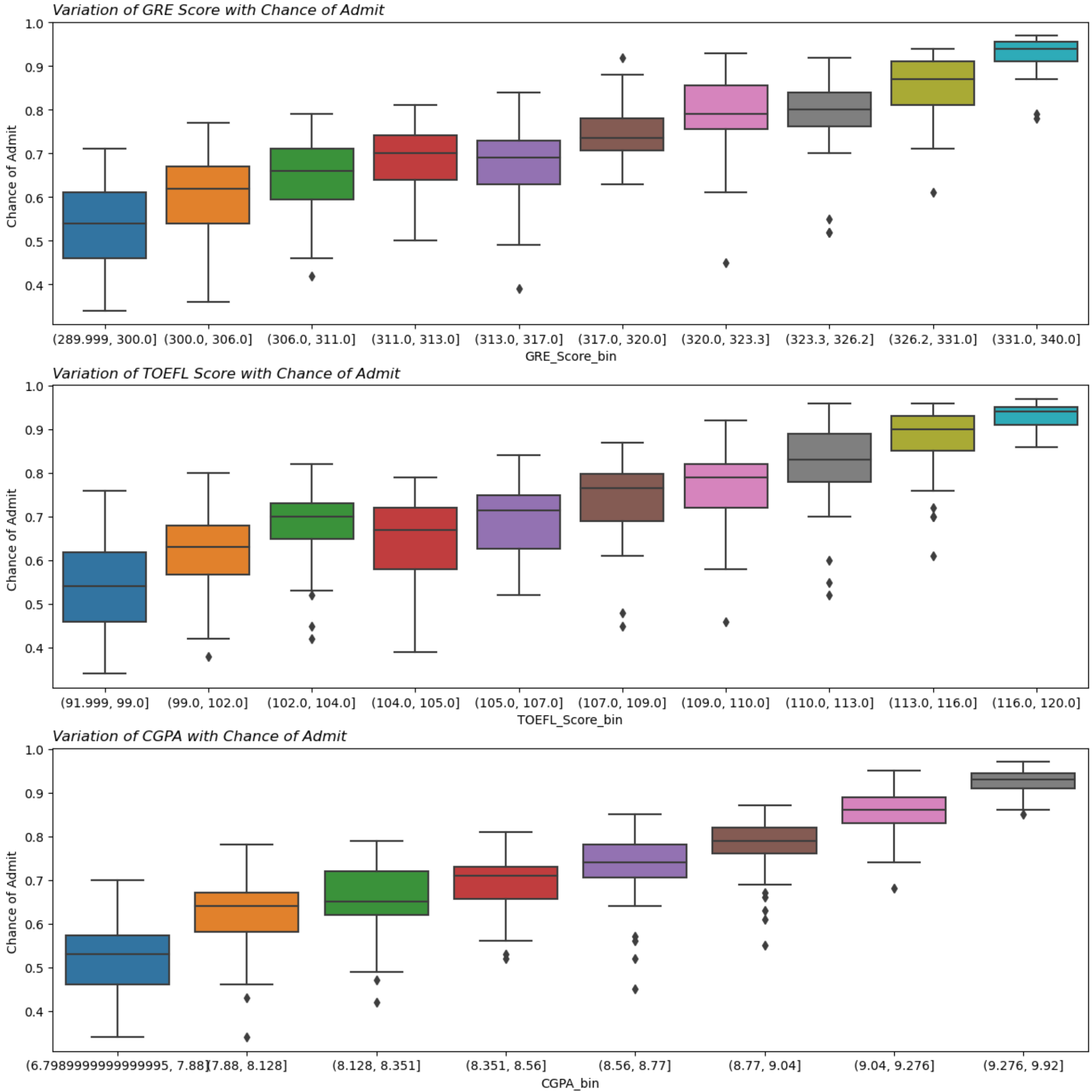
```
fig, axs = plt.subplots(1, 3, figsize=(15,5))
sns.scatterplot(x=df[df_cols_cont[0]], y=df[df_cols_cont[-1]], data=df,ax=axs[0]).
                             df_cols_cont[0]+" on "+df_cols_cont[-1],loc="left",fo
sns.scatterplot(x=df[df_cols_cont[1]], y=df[df_cols_cont[-1]], data=df,ax=axs[1]).
                             df_cols_cont[1]+" on "+df_cols_cont[-1],loc="left",fo
sns.scatterplot(x=df[df_cols_cont[2]], y=df[df_cols_cont[-1]], data=df,ax=axs[2]).
                             df_cols_cont[2]+" on "+df_cols_cont[-1],loc="left",fo
plt.show()
```



Outliers of features having continuous values with respect to chance of Admission

In [125...

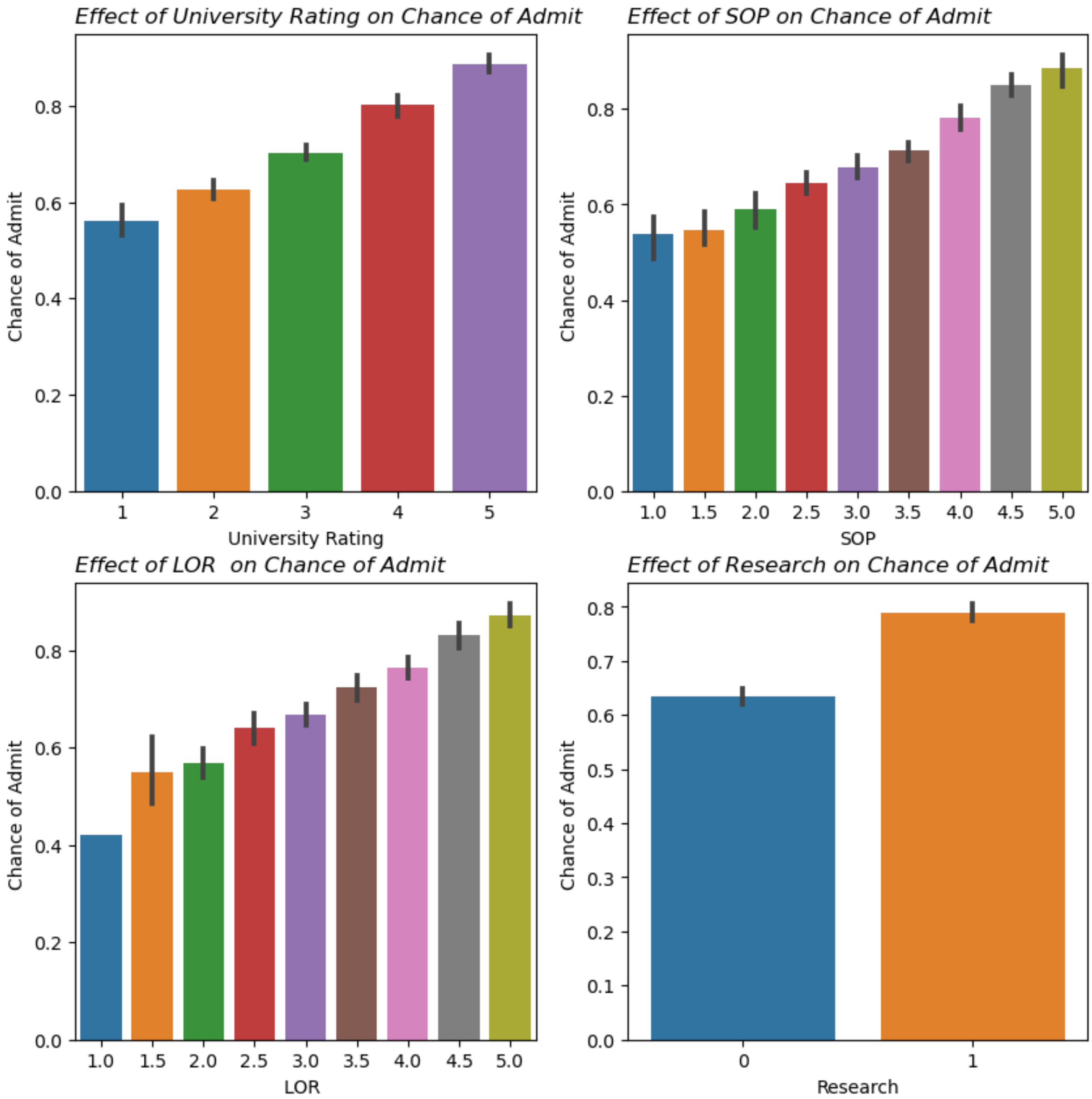
```
df["GRE_Score_bin"]=pd.qcut(df["GRE Score"],10)
df["TOEFL_Score_bin"]=pd.qcut(df["TOEFL Score"],10)
df["CGPA_bin"]=pd.qcut(df["CGPA"],8)
fig, axs = plt.subplots(3, 1, figsize=(15,15))
sns.boxplot(x=df["GRE_Score_bin"], y=df[df_cols_cont[-1]], data=df,ax=axs[0]).set_
                            df_cols_cont[0]+" with "+df_cols_cont[-1],loc="left",f
sns.boxplot(x=df["TOEFL_Score_bin"], y=df[df_cols_cont[-1]], data=df,ax=axs[1]).se
                             df_cols_cont[1]+" with "+df_cols_cont[-1],loc="left",
sns.boxplot(x=df["CGPA_bin"], y=df[df_cols_cont[-1]], data=df,ax=axs[2]).set_title
                             df_cols_cont[2]+" with "+df_cols_cont[-1],loc="left",
plt.show()
```



Bivariate analysis of features having discrete and binary values against the chance of admission

In [126…

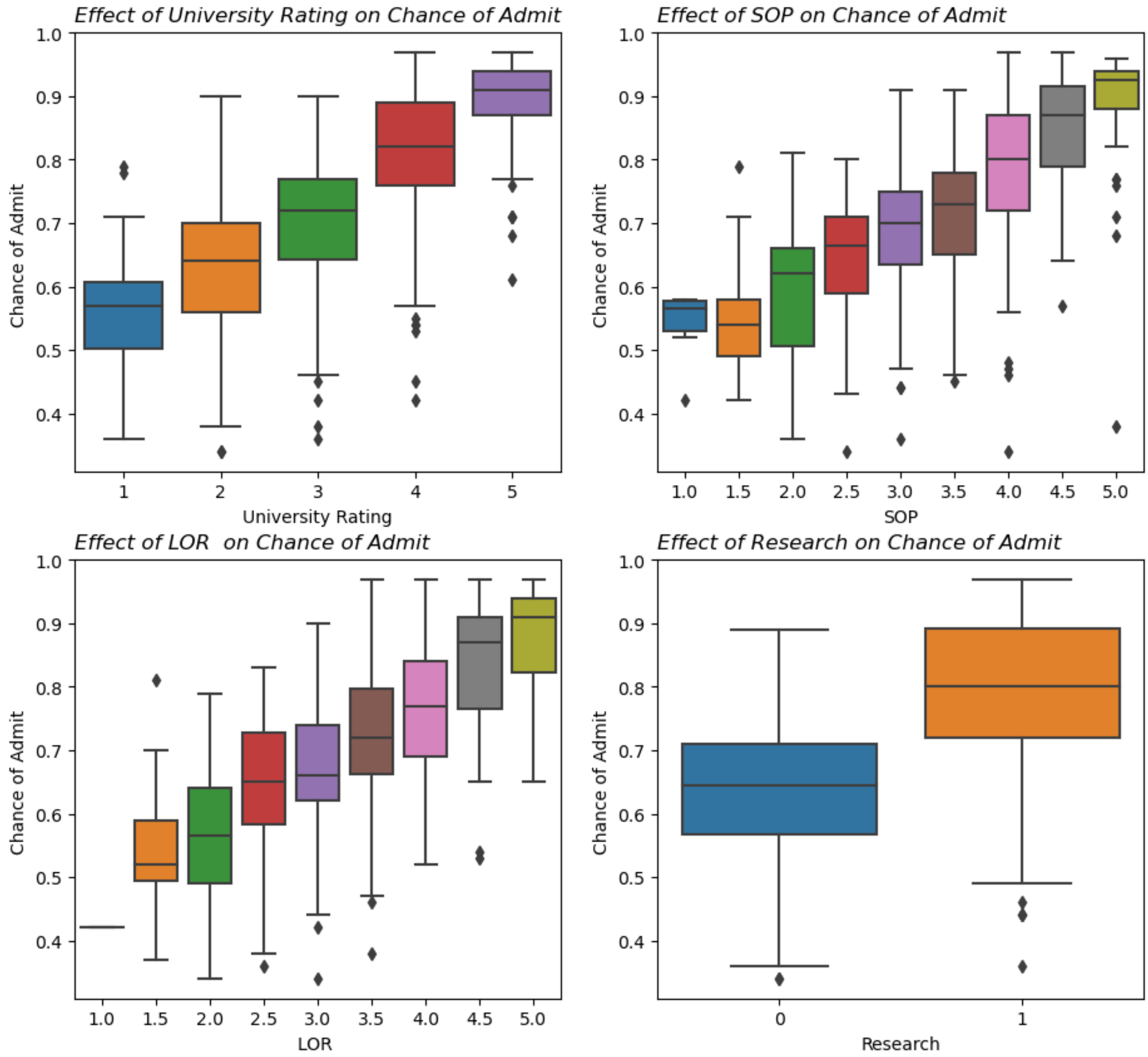
```
fig, axs = plt.subplots(2, 2, figsize=(10,10))
sns.barplot(x=df[df_cols_disc[0]], y=df[df_cols_cont[-1]], data=df,ax=axs[0,0]).se
                             df_cols_disc[0]+" on "+df_cols_cont[-1],loc="left",fo
sns.barplot(x=df[df_cols_disc[1]], y=df[df_cols_cont[-1]], data=df,ax=axs[0,1]).se
                             df_cols_disc[1]+" on "+df_cols_cont[-1],loc="left",fo
sns.barplot(x=df[df_cols_disc[2]], y=df[df_cols_cont[-1]], data=df,ax=axs[1,0]).se
                             df_cols_disc[2]+" on "+df_cols_cont[-1],loc="left",fo
sns.barplot(x=df[df_cols_disc[3]], y=df[df_cols_cont[-1]], data=df,ax=axs[1,1]).se
                             df_cols_disc[3]+" on "+df_cols_cont[-1],loc="left",fo
plt.show()
```



Outliers of features having discrete and binary values with respect to chance of Admission

In [127…

```
fig, axs = plt.subplots(2, 2, figsize=(11,10))
sns.boxplot(x=df[df_cols_disc[0]], y=df[df_cols_cont[-1]], data=df,ax=axs[0,0]).se
                             df_cols_disc[0]+" on "+df_cols_cont[-1],loc="left",fo
sns.boxplot(x=df[df_cols_disc[1]], y=df[df_cols_cont[-1]], data=df,ax=axs[0,1]).se
                             df_cols_disc[1]+" on "+df_cols_cont[-1],loc="left",fo
sns.boxplot(x=df[df_cols_disc[2]], y=df[df_cols_cont[-1]], data=df,ax=axs[1,0]).se
                             df_cols_disc[2]+" on "+df_cols_cont[-1],loc="left",fo
sns.boxplot(x=df[df_cols_disc[3]], y=df[df_cols_cont[-1]], data=df,ax=axs[1,1]).se
                             df_cols_disc[3]+" on "+df_cols_cont[-1],loc="left",fo
plt.show()
```



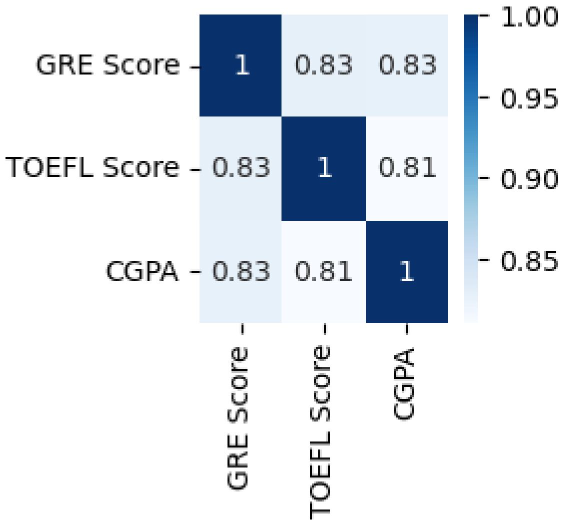
Multivariate Analysis

Multivariate analysis of features

In [128…

```
# Analyze relationship of all features having continuous values
df_cont_corr = df[["GRE Score","TOEFL Score","CGPA"]].corr()
plt.figure(figsize=(2,2))
sns.heatmap(df_cont_corr, annot=True,cmap = "Blues")
display(df_cont_corr)
display(plt.show())
```

	GRE Score	TOEFL Score	CGPA
GRE Score	1.000000	0.827200	0.825878
TOEFL Score	0.827200	1.000000	0.810574
CGPA	0.825878	0.810574	1.000000

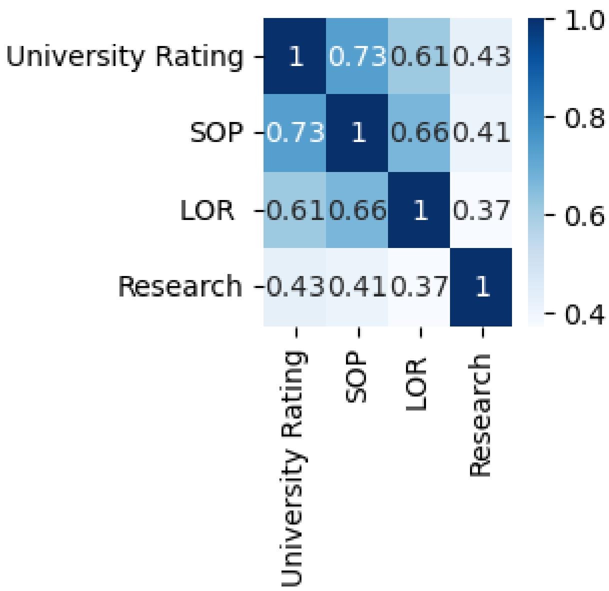


None

In [129…

```
# Analyze relationship of all features having discrete values
df_disc_corr = df[["University Rating","SOP","LOR ","Research"]].corr()
plt.figure(figsize=(2,2))
sns.heatmap(df_disc_corr, annot=True,cmap = "Blues")
display(df_disc_corr)
display(plt.show())
```

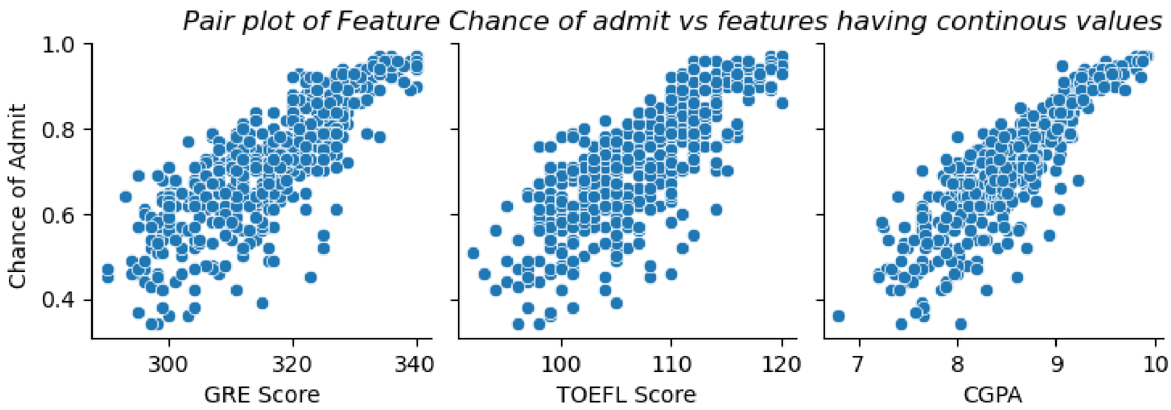
	University Rating	SOP	LOR	Research
University Rating	1.000000	0.728024	0.608651	0.427047
SOP	0.728024	1.000000	0.663707	0.408116
LOR	0.608651	0.663707	1.000000	0.372526
Research	0.427047	0.408116	0.372526	1.000000



None

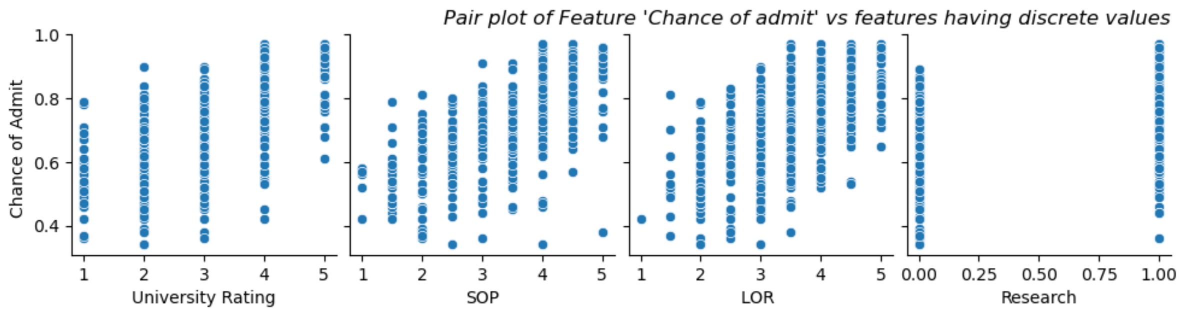
In [130…

```
sns.pairplot(df, x_vars=["GRE Score", "TOEFL Score", "CGPA"],y_vars=["Chance of Adı
plt.title('Pair plot of Feature Chance of admit vs features having continous values
plt.show()
```



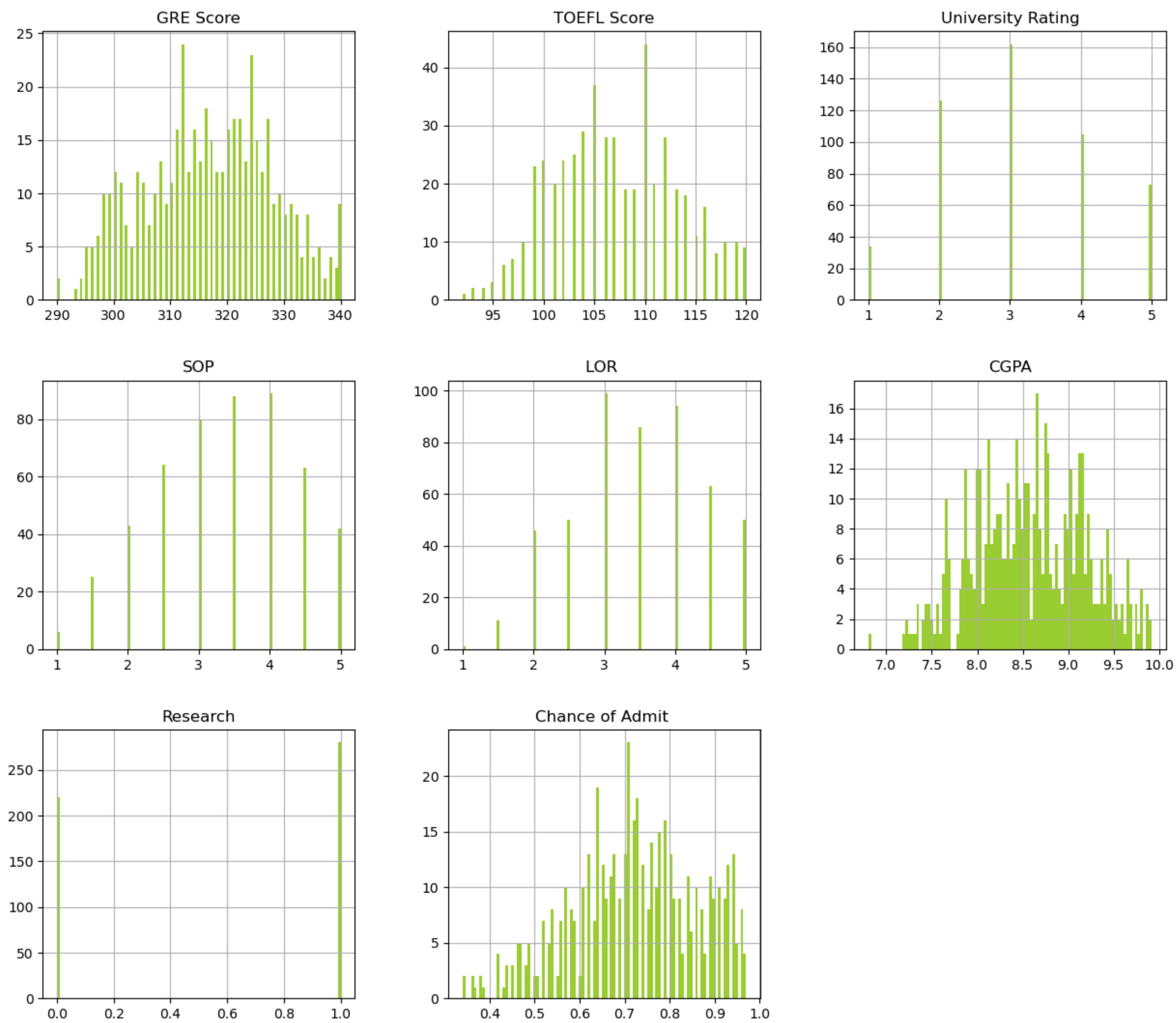
In [131…

```
sns.pairplot(df, x_vars=['University Rating','SOP','LOR ','Research'],y_vars=["Cha
plt.title("Pair plot of Feature 'Chance of admit' vs features having discrete value
plt.show()
```



In [132…

```
df.hist(bins=100,figsize=(15,13),color= "yellowgreen")
plt.show()
```



Data Preprocessing

Duplicate Value Check

In [133…

```
df.duplicated().sum()

# There are no duplicate data points present in dataset
```

Out[133]:

```
0
```

Missing Value treatment

In [134…

```
df.isna().sum().sum()

# There are no missing values in dataset
```

Out[134]:

```
0
```

Outlier Treatment

```
As observed from boxplot (shown above in part 1 of this report) of
different features of dataset,
feature 'LOR' and feature 'Chance of Admit' has outliers.
No outliers present in other features of dataset
```

In [135…

```
#Calculating IQR, upper and Lower bound for LOR feature in dataset
Q1 = np.percentile(df["LOR "], 25,interpolation = 'midpoint')
Q3 = np.percentile(df["LOR "], 75,interpolation = 'midpoint')
IQR = Q3 - Q1
upper=Q3+1.5*IQR
lower=Q1-1.5*IQR
print("upper:",upper,"&","lower:",lower)
display(df[df["LOR "]<lower][['GRE Score','TOEFL Score','University Rating','SOP',
                              'Research','Chance of Admit ']])

# It can be observed that only one outlier is present in 'LOR' feature
```

```
upper: 5.5 & lower: 1.5
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
347	299	94	1	1.0	1.0	7.34	0	0.42

In [136…

```
#Calculating IQR, upper and Lower bound for Chance of Admit in dataset
Q1 = np.percentile(df['Chance of Admit '], 25,interpolation = 'midpoint')
Q3 = np.percentile(df['Chance of Admit '], 75,interpolation = 'midpoint')
IQR = Q3 - Q1
upper=Q3+1.5*IQR
lower=Q1-1.5*IQR
print("upper:",np.round(upper,3),"&","lower:",np.round(lower,3))
display(df[df['Chance of Admit ']<lower][['GRE Score','TOEFL Score','University Ra
                                          'CGPA','Research','Chance of Admit ']])

# It can be observed that two outliers are present in 'Chance of Admit' feature.
```

```
upper: 1.105 & lower: 0.345
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
92	298	98	2	4.0	3.0	8.03	0	0.34
376	297	96	2	2.5	2.0	7.43	0	0.34

As it can be observed from above two calculations:

1. The outlier value of LOR is 1.0 and lower bound is 1.5. The outlier value is not much far from lower bound. Hence,there is no use of deleting that data point.
2. Similarly, the outlier value of 'Chance of Admit' is 0.34 and lower bound is 0.345. Here also, the outlier value is not much far from lower bound and hence, deleting data point will be of no use.
3. The percentage of outliers present in whole dataset is too low to be treated in any way.

Conclusion No data has been deleted as outliers are not affecting dataset much

Feature engineering

In [137…

```
df.head(2)
```

Out[137]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	GRE_Score_bin	TOEFL_Score_bir
0	337	118	4	4.5	4.5	9.65	1	0.92	(331.0, 340.0]	(116.0, 120.0
1	324	107	4	4.0	4.5	8.87	1	0.76	(323.3, 326.2]	(105.0, 107.0

```
As seen above, three columns have been added for purpose of detail
visualization.
These columns are not relevant for model building. Hence, drop
those columns.
```

In [138…

```
data = df.drop(['GRE_Score_bin','TOEFL_Score_bin','CGPA_bin'],axis=1)
```

In [139…

```
data.head(2)
```

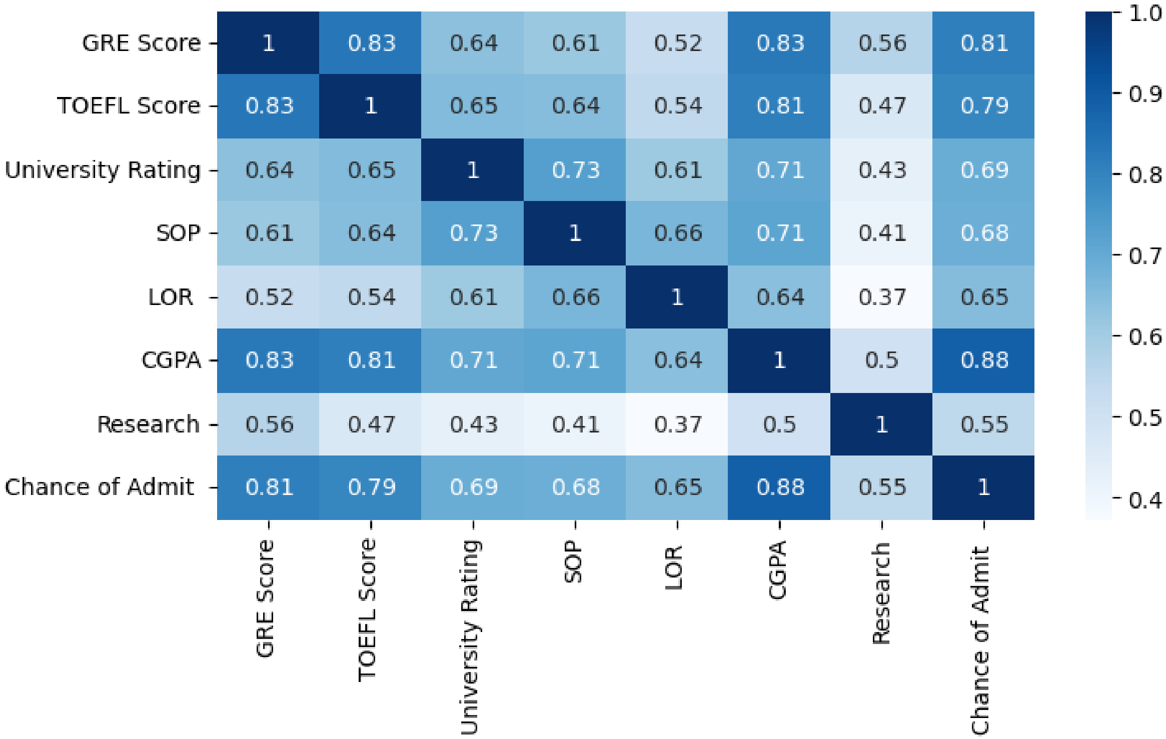
Out[139]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76

```
As values of all features are numerical and within certain range,
there is no need of adding any
new features.
```

correlation among features

```
# Correlation between all features
plt.figure(figsize=(8,4))
sns.heatmap(data.corr(), annot=True,cmap = "Blues")
plt.show()
```



From the above plot,

1. feature 'GRE Score' has maximum relation with features 'TOEFL score' and 'CGPA' followed by feature 'Chance of Admit' in comparison with other features.
2. feature 'TOEFL score' has maximum correlation with features 'GRE Score' and 'CGPA' followed by feature 'Chance of Admit' in comparison with other features.
3. feature 'CGPA' has maximum relation with 'Chance of Admit' followed by features 'GRE Score' and 'TOEFL Score'
4. ***feature 'Chance of Admit' has highest correlation with 'CGPA', followed by 'GRE Score' and 'TOEFL Score'***

Data preparation for modeling

In [141…

```
#Standardising data
# Import library
from  sklearn.preprocessing import MinMaxScaler
```

In [142…

```
scaler = MinMaxScaler()
data = pd.DataFrame(scaler.fit_transform(data), columns=data.columns)
data.head()
```

Out[142]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	0.94	0.928571	0.75	0.875	0.875	0.913462	1.0	0.920635
1	0.68	0.535714	0.75	0.750	0.875	0.663462	1.0	0.666667
2	0.52	0.428571	0.50	0.500	0.625	0.384615	1.0	0.603175
3	0.64	0.642857	0.50	0.625	0.375	0.599359	1.0	0.730159
4	0.48	0.392857	0.25	0.250	0.500	0.451923	0.0	0.492063

In [143…

```
#split dataset into independent and dependent features
X = data.drop(["Chance of Admit "],axis = 1)       # independent variables
y = data["Chance of Admit "].values.reshape(-1,1)  # target/dependent variables
X.shape, y.shape
```

Out[143]:

```
((500, 7), (500, 1))
```

After scaling all the features, split the dataset into training set and test set

In [144…

```
from sklearn.model_selection import train_test_split
```

In [145…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
```

In [146…

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(350, 7) (150, 7) (350, 1) (150, 1)
```

Data has been pre-processed and now ready for linear regression

Model Building

Linear Regression model and model statistics

a) Using statsmodel

In [147…

```
# In general, scikit-learn is designed for prediction, while statsmodels is more su
# The difference between the two libraries is how they handle constants.
# Scikit-learn allows the user to specify whether or not to add a constant through
# while statsmodels’ OLS class has a function that adds a constant to a given array
import statsmodels.api as sm

X_sm = sm.add_constant(X)
sm_model = sm.OLS(y, X_sm).fit()
print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     324.4
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          8.21e-180
Time:                        20:54:46   Log-Likelihood:                 470.37
No. Observations:                 500   AIC:                            -924.7
Df Residuals:                     492   BIC:                            -891.0
Df Model:                           7
Covariance Type:            nonrobust
==================================================================================
===
                        coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
const                 0.0130      0.014      0.902      0.367      -0.015       0.
041
GRE Score             0.1475      0.040      3.700      0.000       0.069       0.
226
TOEFL Score           0.1235      0.039      3.184      0.002       0.047       0.
200
University Rating     0.0377      0.024      1.563      0.119      -0.010       0.
085
SOP                   0.0101      0.029      0.348      0.728      -0.047       0.
067
LOR                   0.1070      0.026      4.074      0.000       0.055       0.
159
CGPA                  0.5863      0.048     12.198      0.000       0.492       0.
681
Research              0.0386      0.010      3.680      0.000       0.018       0.
059
==============================================================================
Omnibus:                      112.770   Durbin-Watson:                   0.796
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              262.104
Skew:                          -1.160   Prob(JB):                     1.22e-57
Kurtosis:                       5.684   Cond. No.                         23.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Insights from the analysis:-

1. This is OLS (Ordinary least squares) model and Least square method is applied.
2. The R-squared value of model is 0.822
3. The adjusted R-squared value of model is 0.819

b) Using Scikit-learn

In [148…

```
#Import LinearRegression from sklearn librar

from sklearn.linear_model import LinearRegression
```

In [149…

```
model1 = LinearRegression()

# train the model
model1.fit(X_train, y_train)
```

```
#predict the model
y_pred = model1.predict(X_test)
```

Displaying model coefficients with column names

In [150…

```
# Find all the coefficients
X_coef = pd.DataFrame(model1.coef_)
X_coeff = pd.DataFrame()
X_coeff["features"] = X.columns
X_coeff["coefficient_simple_LR"] = X_coef.T
X_coeff
```

Out[150]:

	features	coefficient_simple_LR
0	GRE Score	0.177005
1	TOEFL Score	0.152346
2	University Rating	0.019651
3	SOP	0.009619
4	LOR	0.096634
5	CGPA	0.573403
6	Research	0.032936

In [151…

```
# Find which feature is most important
X_test.columns[np.argmax(np.abs(model1.coef_))]
```

Out[151]:

```
'CGPA'
```

In [152…

```
# Find which feature is least important
X_test.columns[np.argmin(np.abs(model1.coef_))]
```

Out[152]:

```
'SOP'
```

1. Features such as CGPA, GRE Score and TOEFL Score have coefficients more than 0.1,which means an increase in these scores lead to maximum chances of admission
2. On the other hand, Features such as University Rating and Research have coefficients are too low, that means, these features affect the chance of admission but not as much as features like CGPA, GRE and TOEFL Score do.
3. CGPA is the most important feature whereas SOP is least important.

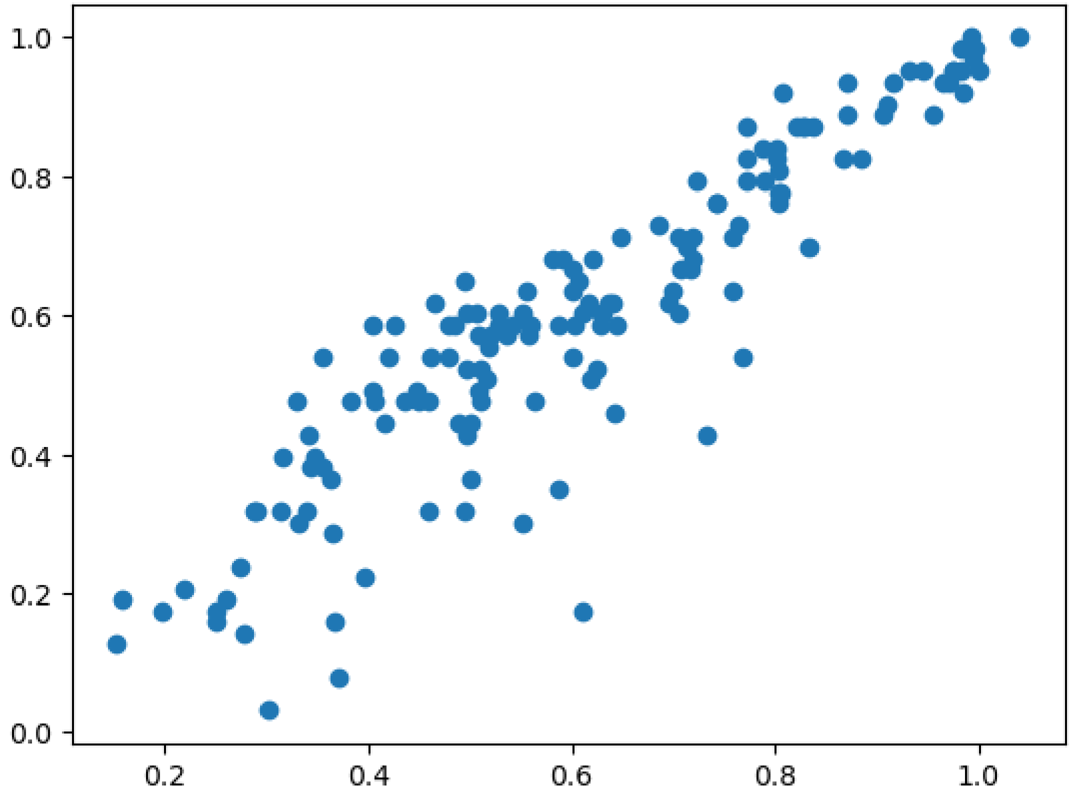
In [153…

```
# Find the intercept
print("Intercept is:", model1.intercept_)
```

```
Intercept is: [0.01133415]
```

In [154…

```
# plot the prediction
fig = plt.figure()
plt.scatter(y_pred,y_test)
plt.show()
```



In the above plot, it can be observed that predicton value and ground truth values follow a bit linear path. However, there are still some outliers, which are not linear. Hence, we need to study each feature and its score and based on that conlcude the important feature.

measure the performance of model1 - Simple Linear Regression Model

In [155…

```
#from statsmodels.stats.outliers_influence import variance_inflation_factor
#from sklearn.feature_selection import f_regression
# Import Libraries
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error, adjus
```

function for Adjusted R2 Score

In [156…

```
def AdjustedR2score(R2,n,d):
    return 1-(((1-R2)*(n-1))/(n-d-1))
```

Calculate Mean Square Error, Root mean square error, Mean absolute error, R2 Score and Adjusted R2 Score of model1

In [157…

```
# Representation
def AdjustedR2score(R2,n,d):
    return 1-(((1-R2)*(n-1))/(n-d-1))
Metrics_reg = pd.DataFrame({ "Metrics simple LR": ["MSE","RMSE","MAE","R2 Score","A
                              "metric value":[mean_squared_error(y_test,y_pred),
                                              np.sqrt(mean_squared_error(y_test,y_
                                              mean_absolute_error(y_test,y_pred),
                                              r2_score(y_test,y_pred),
                                              AdjustedR2score(r2_score(y_test,y_pr
                             })
Metrics_reg
```

Out[157]:

	Metrics simple LR	metric value
0	MSE	0.009157
1	RMSE	0.095690
2	MAE	0.067736
3	R2 Score	0.825631
4	Adjusted R2 Score	0.823150

Regularization - Ridge and Lasso Regression

1. Ridge and Lasso regression are some of the simple techniques to reduce model complexity and prevent over-fitting which may result from simple linear regression.
2. They work by penalizing the magnitude of coefficients of features and minimizing the error between predicted and actual observations. These are called ‘regularization’ techniques. The key difference is in how they assign penalties to the coefficients

Ridge Regression - L2 Regularization

In [158…

```
from sklearn.linear_model import Ridge

model2=Ridge()

#train the model
model2.fit(X_train, y_train)

#predict the model
yr_pred=model2.predict(X_test)
```

In [159…

```
# Find all the coefficients
X_coef_ridge = pd.DataFrame(model2.coef_)
X_coeff_ridge = pd.DataFrame()
X_coeff_ridge["features"] = X.columns
X_coeff_ridge["coefficient_ridge"] = X_coef_ridge.T
X_coeff_ridge
```

Out[159]:

	features	coefficient_ridge
0	GRE Score	0.201448
1	TOEFL Score	0.167972
2	University Rating	0.035235
3	SOP	0.033194
4	LOR	0.104450
5	CGPA	0.451648
6	Research	0.036156

In [160…

```
# Find which feature is most important
X_test.columns[np.argmax(np.abs(model2.coef_))]
```

Out[160]:

```
'CGPA'
```

In [161…

```
# Find which feature is least important
X_test.columns[np.argmin(np.abs(model2.coef_))]
```

Out[161]:

```
'SOP'
```

Here also, the most important feature is CGPA and the least important feature is SOP

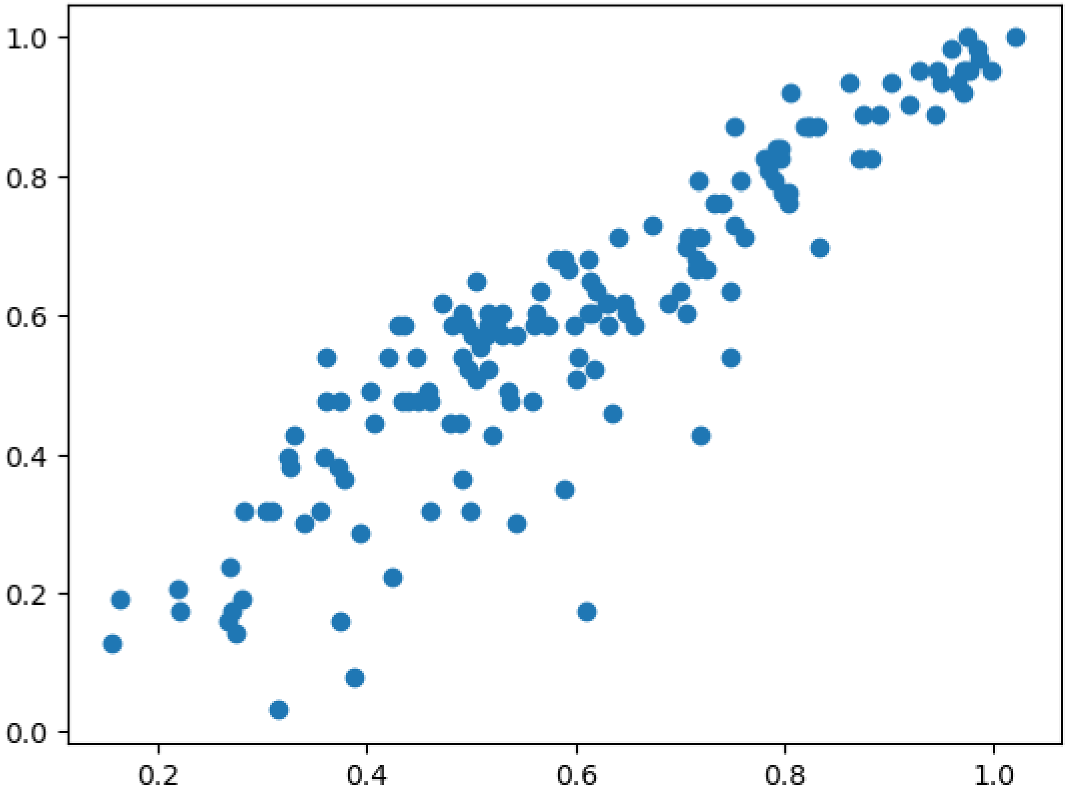
In [162…

```
#Find the intercept
print("Intercept:",model2.intercept_.round(4))
```

```
Intercept: [0.0303]
```

In [163…

```
# plot the prediction
fig = plt.figure()
plt.scatter(yr_pred,y_test)
plt.show()
```



Calculate Mean Square Error, Root mean square error, Mean absolute error, R2 Score and Adjusted R2 Score of model2

In [164…

```
# Representation
Metrics_ridge = pd.DataFrame({ "Metrics ridge": ["MSE","RMSE","MAE","R2 Score","Ad
                              "metric value":[mean_squared_error(y_test,yr_pred),
                                              np.sqrt(mean_squared_error(y_test,yr_
                                              mean_absolute_error(y_test,yr_pred),
                                              r2_score(y_test,yr_pred),
                                              AdjustedR2score(r2_score(y_test,yr_pr
                             })
Metrics_ridge
```

Out[164]:

	Metrics ridge	metric value
0	MSE	0.009218
1	RMSE	0.096013
2	MAE	0.068770
3	R2 Score	0.824451
4	Adjusted R2 Score	0.821953

Lasso Regression - L1 Regression

In [165…

```
from sklearn.linear_model import Lasso

model3=Lasso(alpha = 1)

#train the model
model3.fit(X_train, y_train)

#predict the model
yl_pred=model3.predict(X_test)
```

In [166…

```
# Find all the coefficients
X_coef_lasso = pd.DataFrame(model3.coef_)
X_coeff_lasso = pd.DataFrame()
X_coeff_lasso["features"] = X.columns
X_coeff_lasso["coefficient_lasso"] = X_coef_lasso
X_coeff_lasso
```

Out[166]:

	features	coefficient_lasso
0	GRE Score	0.0
1	TOEFL Score	0.0
2	University Rating	0.0
3	SOP	0.0
4	LOR	0.0
5	CGPA	0.0
6	Research	0.0

In [167…

```
# Find which feature is most important
X_test.columns[np.argmax(np.abs(model3.coef_))]
```

Out[167]:

```
'GRE Score'
```

In [168…

```
# Find which feature is least important
X_test.columns[np.argmin(np.abs(model3.coef_))]
```

Out[168]:

```
'GRE Score'
```

As the coefficients of all features is 0 in this model. It is impossible to find out the most and least important feature from this model. Hence, this model is not much of use.

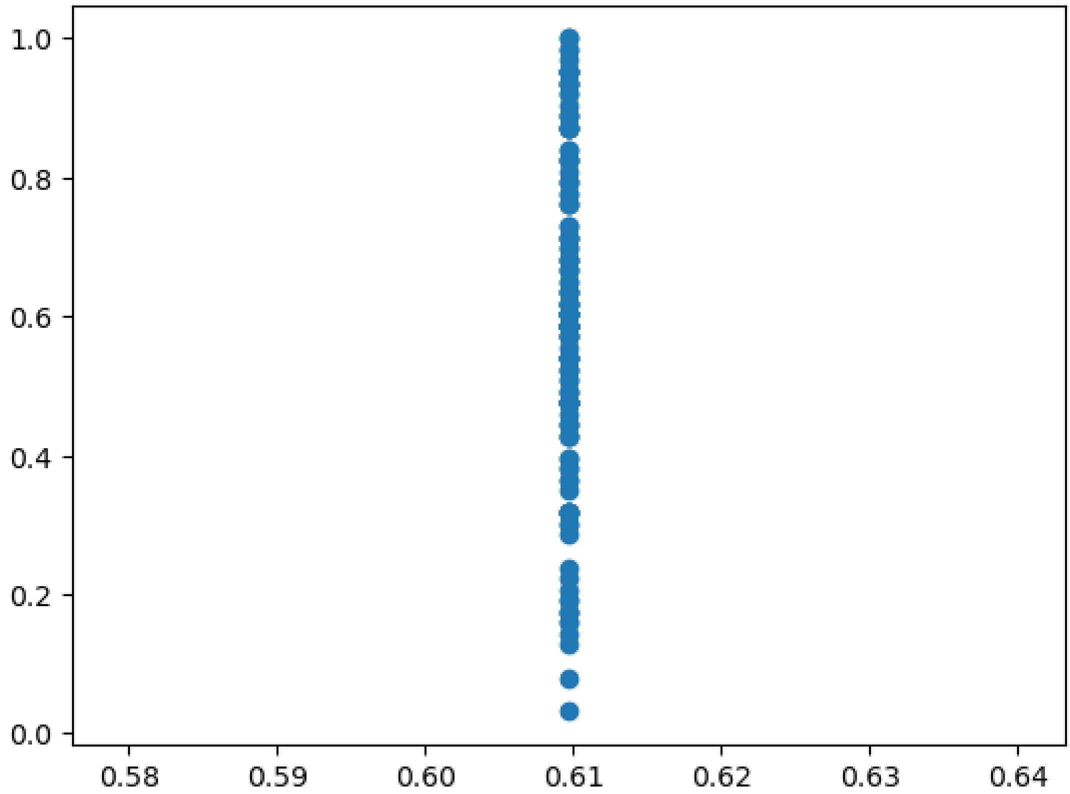
In [169…

```
#Find the intercept
print("Intercept:",model3.intercept_.round(4))
```

```
Intercept: [0.6097]
```

In [170…

```
# plot the prediction
fig = plt.figure()
plt.scatter(yl_pred,y_test)
plt.show()
```



Calculate Mean Square Error, Root mean square error, Mean absolute error, R2 Score and Adjusted R2 Score of model2

In [171…

```
# Representation
Metrics_lasso = pd.DataFrame({ "Metrics_lasso": ["MSE","RMSE","MAE","R2 Score","Ad
                              "metric value":[mean_squared_error(y_test,yl_pred),
                                              np.sqrt(mean_squared_error(y_test,yl_
                                              mean_absolute_error(y_test,yl_pred),
                                              r2_score(y_test,yl_pred),
                                              AdjustedR2score(r2_score(y_test,yl_pr
                             })
Metrics_lasso
```

Out[171]:

	Metrics_lasso	metric value
0	MSE	0.052666
1	RMSE	0.229491
2	MAE	0.182067
3	R2 Score	-0.002933
4	Adjusted R2 Score	-0.017203

Model performance evaluation

Performance and metrics comparison for all three models

In [172…

```
# compare coefficients of all models
display(X_coeff)
display(X_coeff_ridge)
display(X_coeff_lasso)
```

	features	coefficient_simple_LR
0	GRE Score	0.177005
1	TOEFL Score	0.152346
2	University Rating	0.019651
3	SOP	0.009619
4	LOR	0.096634
5	CGPA	0.573403
6	Research	0.032936

	features	coefficient_ridge
0	GRE Score	0.201448
1	TOEFL Score	0.167972
2	University Rating	0.035235
3	SOP	0.033194
4	LOR	0.104450
5	CGPA	0.451648
6	Research	0.036156

	features	coefficient_lasso
0	GRE Score	0.0
1	TOEFL Score	0.0
2	University Rating	0.0
3	SOP	0.0
4	LOR	0.0
5	CGPA	0.0
6	Research	0.0

Comments on coefficients of all models

1. For all models, "GRE Score" feature is most important feature with different values of coefficients.
2. Coefficients values for simple linear regression model and ridge model is almost similar.
3. Coefficients for all features are too small to be considered as 0.

In [173…

```
# compare intercepts of all models
# The intercept of a linear regression model is the value of the dependent variabl
```

```
# independent variables are equal to zero. It can be interpreted as the expected va
# dependent variable when there is no influence from the independent variables.
intercepts_models = pd.DataFrame({ "Model": ["model 1 (simple linear regression)",
                              "intercept value":[model1.intercept_.round(4),
                                                 model2.intercept_.round(4),
                                                 model3.intercept_.round(4)]
                             })
intercepts_models
```

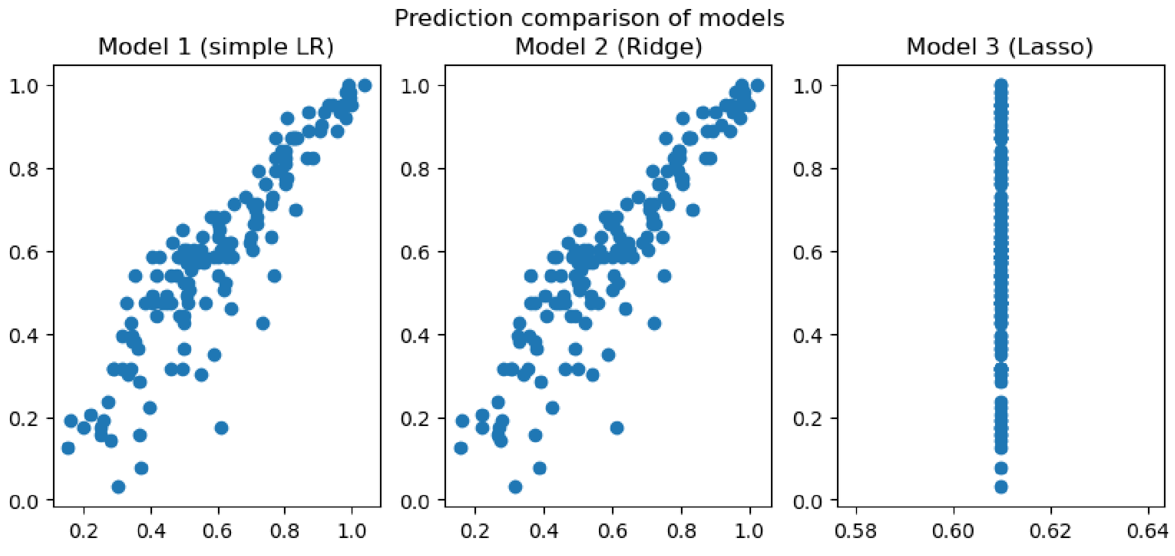
Out[173]:

	Model	intercept value
0	model 1 (simple linear regression)	[0.0113]
1	model 2 (ridge)	[0.0303]
2	model 3 (lasso)	[0.6097]

```
 The intercept of lasso model is high compared to simple and ridge
model.
```

In [174…

```
# Compare the prediction
fig, (ax1, ax2, ax3) = plt.subplots(1,3,figsize=(10, 4))
fig.suptitle('Prediction comparison of models')
ax1.scatter(y_pred, y_test)
ax1.set_title("Model 1 (simple LR)")
ax2.scatter(yr_pred, y_test)
ax2.set_title("Model 2 (Ridge)")
ax3.scatter(yl_pred, y_test)
ax3.set_title("Model 3 (Lasso)")
plt.show()
```



The plots of simple linear regression model and after regression (ridge) model are almost similar. There is no need of applying regularization technique.

In [175…

```
# Comparison of all metrics of all three models
display(Metrics_reg)
display(Metrics_ridge)
display(Metrics_lasso)
```

	Metrics simple LR	metric value
0	MSE	0.009157
1	RMSE	0.095690
2	MAE	0.067736
3	R2 Score	0.825631
4	Adjusted R2 Score	0.823150

	Metrics ridge	metric value
0	MSE	0.009218
1	RMSE	0.096013
2	MAE	0.068770
3	R2 Score	0.824451
4	Adjusted R2 Score	0.821953

	Metrics_lasso	metric value
0	MSE	0.052666
1	RMSE	0.229491
2	MAE	0.182067
3	R2 Score	-0.002933
4	Adjusted R2 Score	-0.017203

comment on the model statistics

1. All errors(MSE,RMSE,MAE) of simple LR model and ridge model are almost same.
2. The R2 and adjusted R2 score of simple LR model and ridge model is almost equal.
3. The metric value of lasso model is quite different from other two models.
4. The simple Linear regression model, here, does not need any regularization technique.

Train and test performances are checked

In [176…

```
predict_train=model1.predict(X_train)
predict_test=model1.predict(X_test)

train_test_performance = pd.DataFrame({ "Metrics": ["r2_score","mean_squared_error
              "train_data":[r2_score(y_train, predict_train).round(3),
                            mean_squared_error(y_train, predict_train).round(3),
                           mean_absolute_error(y_train, predict_train).round(3)],
              "test_data":[r2_score(y_test, predict_test).round(3),
                           mean_squared_error(y_test, predict_test).round(3),
                          mean_absolute_error(y_test, predict_test).round(3)]
             })
train_test_performance
```

Out[176]:

	Metrics	train_data	test_data
0	r2_score	0.818	0.826
1	mean_squared_error	0.009	0.009
2	mean_absolute_error	0.067	0.068

The R2 score, MSE and MAE of model for train data and test data are almost equal.

This concludes that the model build on train data is working fine for test data also.

Testing the assumptions of the linear regression model

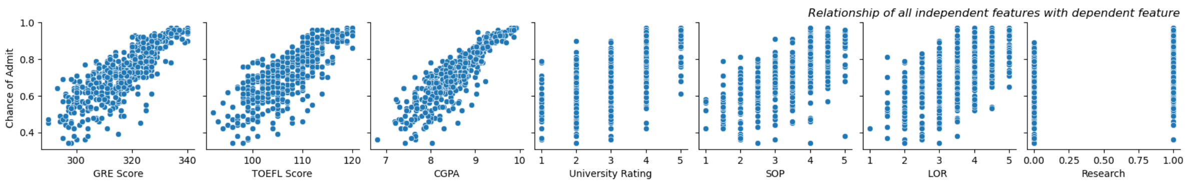
1. Linearity of variables

Linear relationship

1. There should be a linear relationship between the independent variables (inputs) and the dependent variable (output).
2. The change in the dependent variable should be directly proportional to the change in the independent variables. ***Straight Line Fit:***
3. The relationship should be best represented by a straight line.
4. The linear regression model assumes that the relationship between the variables can be expressed using a linear equation, where the coefficients represent the slope of the line.

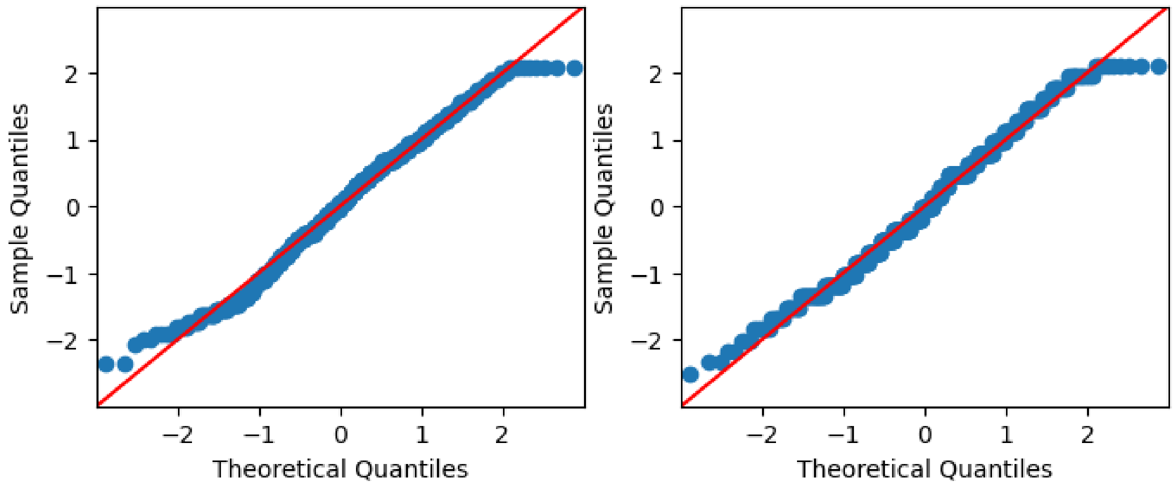
In [177…

```
# Check linearity from graphs
sns.pairplot(df, x_vars=["GRE Score", "TOEFL Score", "CGPA",'University Rating','S
                y_vars=["Chance of Admit "])
plt.title('Relationship of all independent features with dependent feature', loc='
plt.show()
```

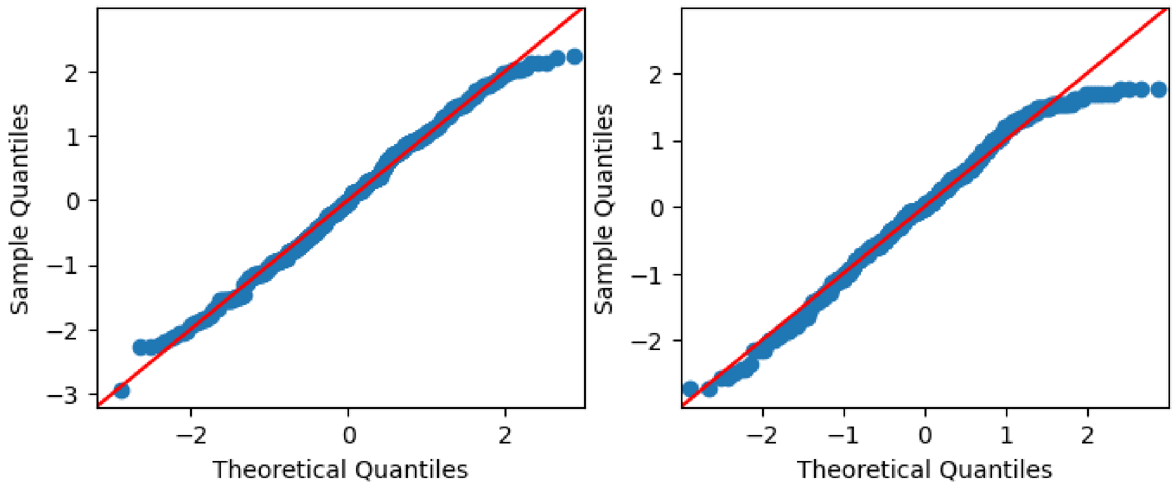


In [178…

```
# Check straight line fit
import statsmodels.api as sm

fig, axs = plt.subplots(1, 2, figsize=(8,3))
sm.qqplot(data["GRE Score"],fit=True, line="45",ax=axs[0])
sm.qqplot(data["TOEFL Score"],fit=True, line="45",ax=axs[1])
plt.show()
```



In [179…
```
fig, axs = plt.subplots(1, 2, figsize=(8,3))
sm.qqplot(data["CGPA"],fit=True, line="45",ax=axs[0])
sm.qqplot(data["Chance of Admit "],fit=True, line="45",ax=axs[1])
plt.show()
```




```
It can be observed that all features having continuous values
follow normal distribution.
Next, split dataset into independent and dependent features
```

2. Multicollinearity Check

1. Multicollinearity can be said as collinearity across multiple features.
2. VIF (Variance Inflation Factor) is used to remove highly correlated features
3. A rule of thumb we can follow: a) *VIF > 10: Very high multicollinearity, drop* b) *5<=VIF<=10: High multicollinearity* c) *VIF<5: Low multicollinearity*

In [180…
```
#VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [181…
```
# calculating VIF
vif = pd.DataFrame()
X_t = X
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1]
vif['VIF'] = round(vif['VIF'], 2)
```

```
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[181]:

	Features	VIF
5	CGPA	41.46
0	GRE Score	29.02
1	TOEFL Score	28.12
3	SOP	19.01
4	LOR	15.05
2	University Rating	11.10
6	Research	3.34

VIF values tends to be infinity when there is a perfect correlation between the variables

Remove the variable with highest VIF

In [182…

```
# Removing feature with highest VIF
cols2 = vif["Features"][1:].values
X2 = X[cols2]

X2_sm = sm.add_constant(X2)
sm_model1 = sm.OLS(y, X2_sm).fit()
print(sm_model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.768
Model:                            OLS   Adj. R-squared:                  0.765
Method:                 Least Squares   F-statistic:                     272.1
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          6.84e-153
Time:                        20:54:48   Log-Likelihood:                 404.31
No. Observations:                 500   AIC:                            -794.6
Df Residuals:                     493   BIC:                            -765.1
Df Model:                           6
Covariance Type:            nonrobust
==================================================================================
===
                        coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
const                 0.0689      0.016      4.405      0.000       0.038       0.
100
GRE Score             0.3402      0.042      8.151      0.000       0.258       0.
422
TOEFL Score           0.2567      0.042      6.053      0.000       0.173       0.
340
SOP                   0.0744      0.032      2.292      0.022       0.011       0.
138
LOR                   0.1739      0.029      5.938      0.000       0.116       0.
231
University Rating     0.0797      0.027      2.927      0.004       0.026       0.
133
Research              0.0417      0.012      3.487      0.001       0.018       0.
065
==============================================================================
Omnibus:                       86.496   Durbin-Watson:                   0.820
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              155.545
Skew:                          -1.006   Prob(JB):                     1.67e-34
Kurtosis:                       4.848   Cond. No.                         19.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [183…

```
# calculate VIF of all features again
vif = pd.DataFrame()
X_t = X[cols2]
vif['Features'] = cols2
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[183]:

	Features	VIF
1	TOEFL Score	25.15
0	GRE Score	24.11
2	SOP	18.09
3	LOR	13.28
4	University Rating	11.02
5	Research	3.34

In [184…

```
# let's remove one more feature
cols2 = vif["Features"][1:].values
X2 = X[cols2]

X2_sm = sm.add_constant(X2)
sm_model1 = sm.OLS(y, X2_sm).fit()
print(sm_model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.751
Model:                            OLS   Adj. R-squared:                  0.748
Method:                 Least Squares   F-statistic:                     297.7
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          1.70e-146
Time:                        20:54:49   Log-Likelihood:                 386.39
No. Observations:                 500   AIC:                            -760.8
Df Residuals:                     494   BIC:                            -735.5
Df Model:                           5
Covariance Type:            nonrobust
==================================================================================
===
                        coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
const                 0.0867      0.016      5.449      0.000       0.055       0.
118
GRE Score             0.5040      0.033     15.322      0.000       0.439       0.
569
SOP                   0.1052      0.033      3.167      0.002       0.040       0.
170
LOR                   0.1844      0.030      6.092      0.000       0.125       0.
244
University Rating     0.1019      0.028      3.648      0.000       0.047       0.
157
Research              0.0386      0.012      3.121      0.002       0.014       0.
063
==============================================================================
Omnibus:                       79.499   Durbin-Watson:                   0.846
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              139.726
Skew:                          -0.943   Prob(JB):                     4.56e-31
Kurtosis:                       4.775   Cond. No.                         12.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [185…

```
# calculate VIF of all features again
vif = pd.DataFrame()
X_t = X[cols2]
vif['Features'] = cols2
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[185]:

	Features	VIF
1	SOP	17.43
2	LOR	12.97
0	GRE Score	12.66
3	University Rating	10.90
4	Research	3.33

In [186…

```
# let's remove one more feature
cols2 = vif["Features"][1:].values
X2 = X[cols2]

X2_sm = sm.add_constant(X2)
sm_model1 = sm.OLS(y, X2_sm).fit()
print(sm_model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.746
Model:                            OLS   Adj. R-squared:                  0.744
Method:                 Least Squares   F-statistic:                     363.0
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          1.19e-145
Time:                        20:54:49   Log-Likelihood:                 381.36
No. Observations:                 500   AIC:                            -752.7
Df Residuals:                     495   BIC:                            -731.7
Df Model:                           4
Covariance Type:            nonrobust
==================================================================================
===
                        coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
const                 0.0960      0.016      6.086      0.000       0.065       0.
127
LOR                   0.2193      0.028      7.706      0.000       0.163       0.
275
GRE Score             0.5241      0.033     16.095      0.000       0.460       0.
588
University Rating     0.1405      0.025      5.533      0.000       0.091       0.
190
Research              0.0393      0.012      3.152      0.002       0.015       0.
064
==============================================================================
Omnibus:                       70.658   Durbin-Watson:                   0.881
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              117.085
Skew:                          -0.875   Prob(JB):                     3.76e-26
Kurtosis:                       4.599   Cond. No.                         11.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [187…

```
# calculate VIF of all features again
vif = pd.DataFrame()
X_t = X[cols2]
vif['Features'] = cols2
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1]
vif['VIF'] = round(vif['VIF'], 2)
```

```
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[187]:

	Features	VIF
1	GRE Score	11.71
0	LOR	9.73
2	University Rating	8.98
3	Research	3.33

In [188…

```
# let's remove one more feature
cols2 = vif["Features"][1:].values
X2 = X[cols2]

X2_sm = sm.add_constant(X2)
sm_model1 = sm.OLS(y, X2_sm).fit()
print(sm_model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.613
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     261.5
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          9.74e-102
Time:                        20:54:49   Log-Likelihood:                 276.13
No. Observations:                 500   AIC:                            -544.3
Df Residuals:                     496   BIC:                            -527.4
Df Model:                           3
Covariance Type:            nonrobust
==================================================================================
===
                        coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
const                 0.1916      0.018     10.638      0.000       0.156       0.
227
LOR                   0.3008      0.035      8.710      0.000       0.233       0.
369
University Rating     0.3042      0.029     10.606      0.000       0.248       0.
361
Research              0.1192      0.014      8.449      0.000       0.091       0.
147
==============================================================================
Omnibus:                       47.057   Durbin-Watson:                   0.925
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               61.205
Skew:                          -0.729   Prob(JB):                     5.12e-14
Kurtosis:                       3.901   Cond. No.                         9.39
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [189…

```
# calculate VIF of all features again
vif = pd.DataFrame()
X_t = X[cols2]
vif['Features'] = cols2
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1]
vif['VIF'] = round(vif['VIF'], 2)
```

```
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[189]:

	Features	VIF
1	University Rating	7.54
0	LOR	7.36
2	Research	2.85

In [190…

```
# let's remove one more feature
cols2 = vif["Features"][1:].values
X2 = X[cols2]

X2_sm = sm.add_constant(X2)
sm_model1 = sm.OLS(y, X2_sm).fit()
print(sm_model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.525
Model:                            OLS   Adj. R-squared:                  0.523
Method:                 Least Squares   F-statistic:                     274.5
Date:                Thu, 24 Aug 2023   Prob (F-statistic):           4.97e-81
Time:                        20:54:49   Log-Likelihood:                 225.04
No. Observations:                 500   AIC:                            -444.1
Df Residuals:                     497   BIC:                            -431.4
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2078      0.020     10.460      0.000       0.169       0.247
LOR            0.4970      0.032     15.404      0.000       0.434       0.560
Research       0.1599      0.015     10.645      0.000       0.130       0.189
==============================================================================
Omnibus:                       30.067   Durbin-Watson:                   1.056
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               33.799
Skew:                          -0.615   Prob(JB):                     4.58e-08
Kurtosis:                       3.331   Cond. No.                         7.14
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [191…

```
# calculate VIF of all features again
vif = pd.DataFrame()
X_t = X[cols2]
vif['Features'] = cols2
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[191]:

	Features	VIF
0	LOR	2.63
1	Research	2.63

Observations:

1. CGPA is highly correlated feature.
2. After removing CGPA, still there are features having VIF greater than 10 and TOEFL Score is highly correlated feature among all.
3. On removing TOEFL Score, R2 and Adjusted R2 score has been decreased and SOP feature is highly correlated among remaining features.
4. After removing SOP, GRE Score feature is highly correlated among remaining features. The R2 and Adjusted R2 Score is approx same.
5. After removing GRE Score, all VIFs are now less than 10. They have high collinearity.
6. Therefore, it can be observed that Research feature and LOR feature is not much correlated. Even University Rating feature is not much correlated too.
7. The probability of chance of admission is highly depend on those features which are highly correlated.

3. mean of residuals is nearly zero

In [192…

```
X_sm = sm.add_constant(X)
sm_model = sm.OLS(y, X_sm).fit()
print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     324.4
Date:                Thu, 24 Aug 2023   Prob (F-statistic):          8.21e-180
Time:                        20:54:49   Log-Likelihood:                 470.37
No. Observations:                 500   AIC:                            -924.7
Df Residuals:                     492   BIC:                            -891.0
Df Model:                           7
Covariance Type:            nonrobust
==================================================================================
===
                        coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
const                 0.0130      0.014      0.902      0.367      -0.015       0.
041
GRE Score             0.1475      0.040      3.700      0.000       0.069       0.
226
TOEFL Score           0.1235      0.039      3.184      0.002       0.047       0.
200
University Rating     0.0377      0.024      1.563      0.119      -0.010       0.
085
SOP                   0.0101      0.029      0.348      0.728      -0.047       0.
067
LOR                   0.1070      0.026      4.074      0.000       0.055       0.
159
CGPA                  0.5863      0.048     12.198      0.000       0.492       0.
681
Research              0.0386      0.010      3.680      0.000       0.018       0.
059
==============================================================================
Omnibus:                      112.770   Durbin-Watson:                   0.796
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              262.104
Skew:                          -1.160   Prob(JB):                     1.22e-57
Kurtosis:                       5.684   Cond. No.                         23.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [193…

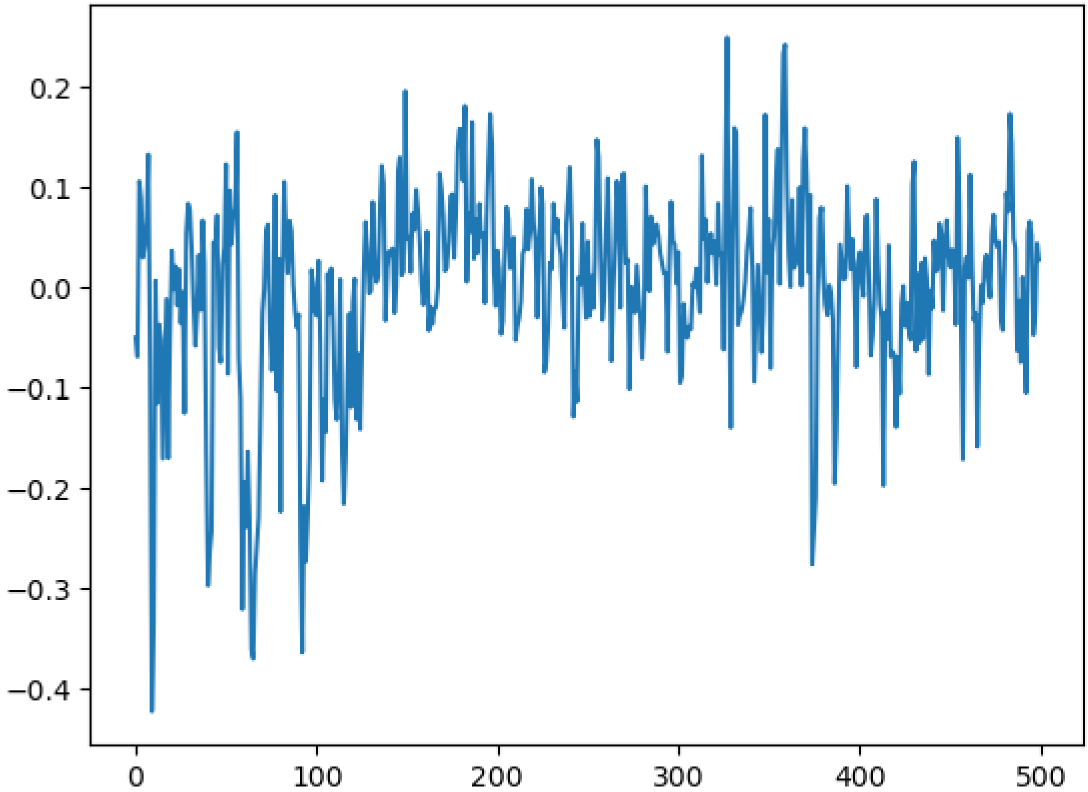
```python
# Check residuals of model
residuals=pd.DataFrame(sm_model.resid)
residuals.rename(columns = {0:'residual error'}, inplace = True)

# lets set threshold as 0.5 and -0.5 for finding errors which are not close to 0
display(residuals[(residuals["residual error"]>0.5) | (residuals["residual error"]
display(residuals[(residuals["residual error"]>0.4) | (residuals["residual error"]

# The errors of all datapoints are nearly 0
# No datapoints have erorr more than absolute value of 0.5
# Only one datapoint is having error more than absolute value of 0.4
```

	residual error

	residual error
9	-0.423265

In [194…

```
#Plot graph for residuals for original model
plt.plot(residuals.index,residuals)
plt.show()

# The errors are from -0.4 to 0.2.
```



In [195…

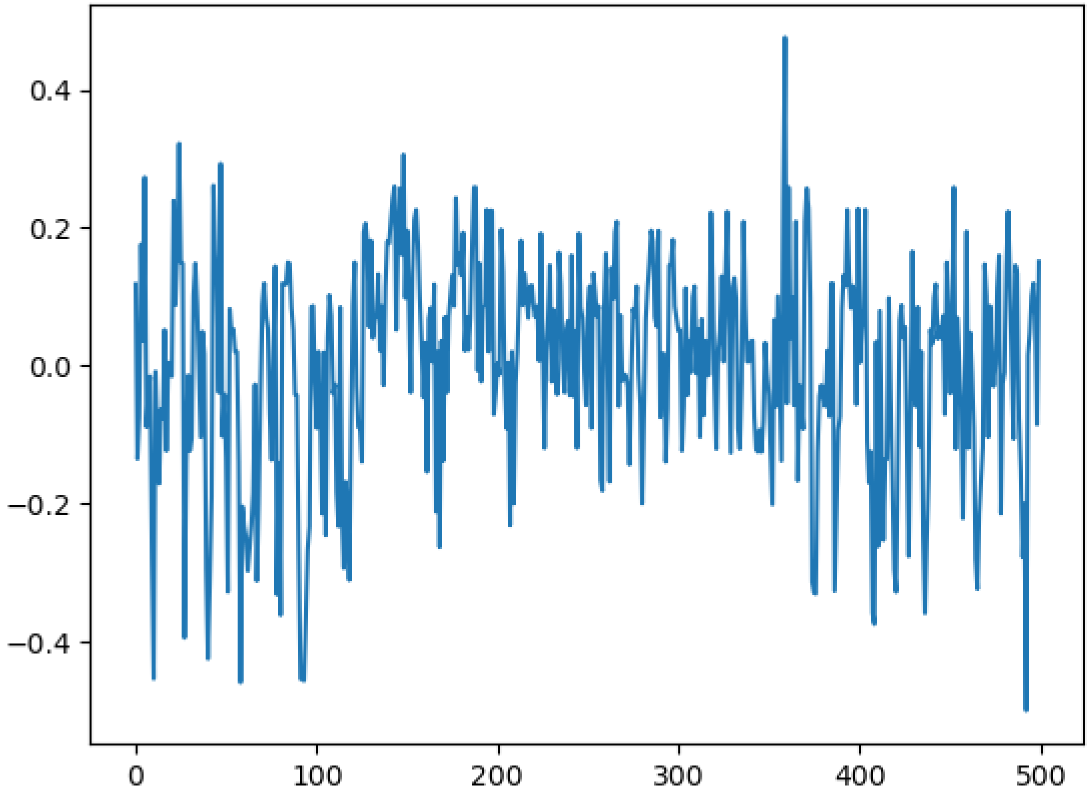
```
#check residual of model1 (after removing features)
residuals1=pd.DataFrame(sm_model1.resid)
residuals1.rename(columns = {0:'residual error'}, inplace = True)

# lets set threshold as 0.5 and -0.5 for finding errors which are not close to 0
display(residuals1[(residuals1["residual error"]>0.5) | (residuals1["residual erro
display(residuals1[(residuals1["residual error"]>0.4) | (residuals1["residual erro

# The errors of all datapoints are nearly 0
# One datapoint is having erorr more than absolute value of 0.5
# 8 datapoints are having error more than absolute value of 0.4
```

	residual error
492	-0.500944

	residual error
10	-0.454695
40	-0.425688
58	-0.460174
91	-0.454880
92	-0.456250
93	-0.457434
359	0.476149
492	-0.500944

In [196...

```
#Plot graph for residuals of model after removing highly correlated features
plt.plot(residuals1.index,residuals1)
plt.show()
# The errors are from -0.5 to 0.45.
```



Errors are more in model1 compare to original model. However, in both models, mean of residuals is nearly zero

4. Normality of residuals

The assumption of multivariate normality in linear regression states:

1. The residuals or errors of the regression model follow a multivariate normal distribution.
2. In other words, the errors should be jointly normally distributed across all levels of the independent variables.
3. They should be symmetric and bell-shaped when plotted against their predicted values.

In [197…

```
X_sm = sm.add_constant(X)
sm_model = sm.OLS(y, X_sm).fit()
y_hat = pd.DataFrame(sm_model.predict())
error = y_hat - y
error
```

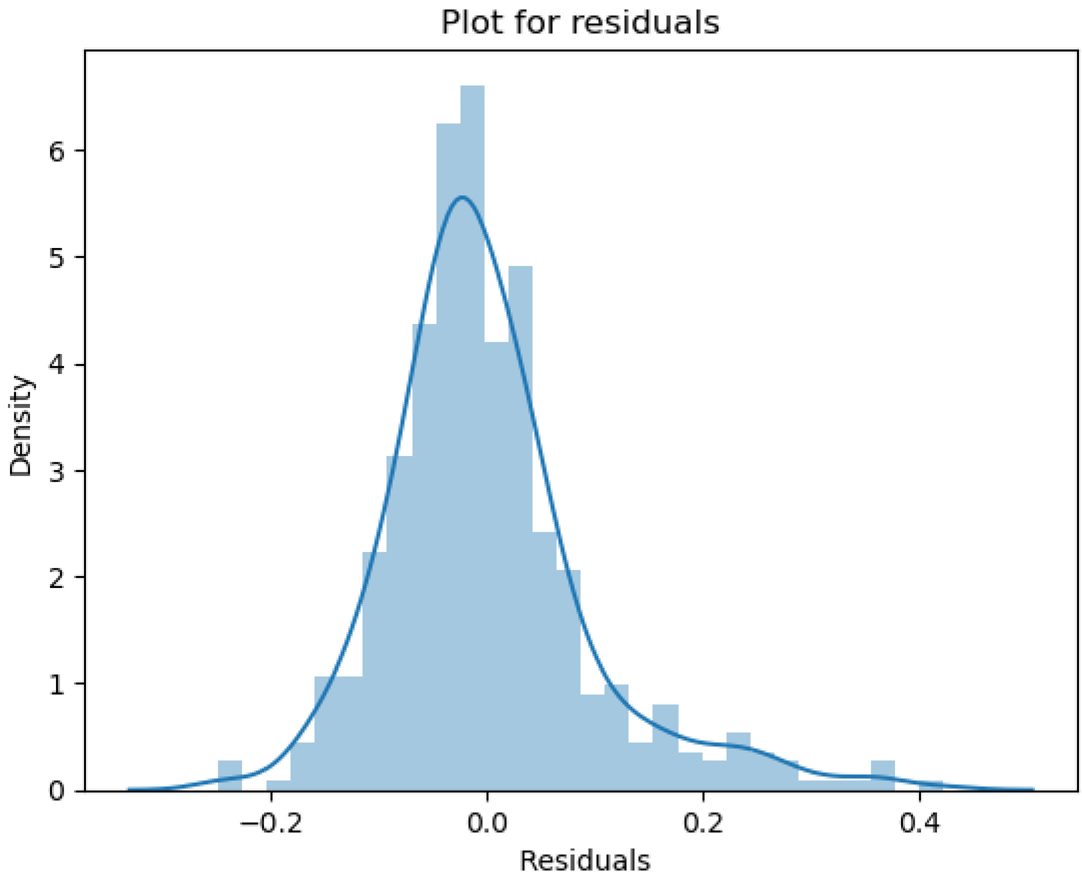
Out[197]:

	0
0	0.050608
1	0.069891
2	-0.105638
3	-0.088065
4	-0.029286
...	...
495	-0.051206
496	0.048117
497	0.028802
498	-0.043413
499	-0.027166

500 rows × 1 columns

In [198…

```
sns.distplot(error)
plt.xlabel(" Residuals")
plt.title("Plot for residuals")
plt.show()
```



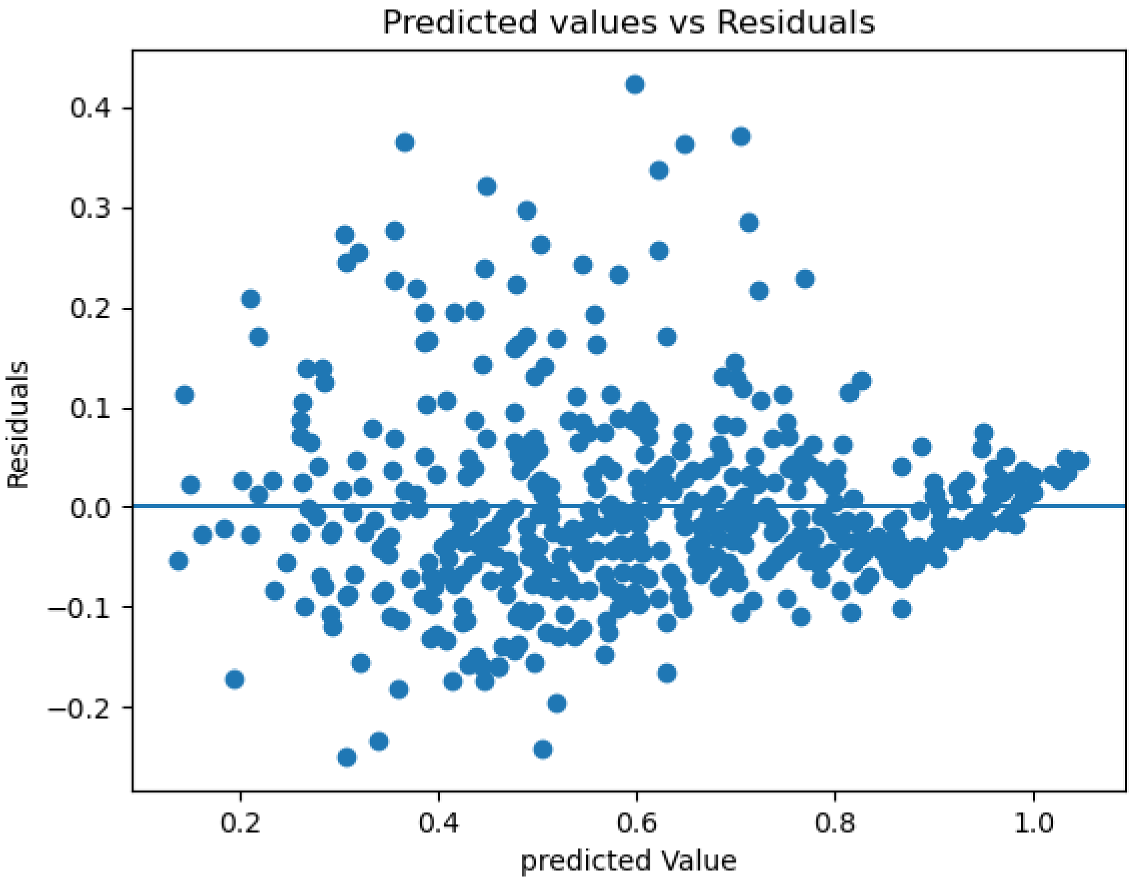
The distribution of errors is gaussian distribution hence the assumption of error being normally distributed is being sastified

4. Test for Homoscedasticity

Errors must be uniform. It should not be increasing with predicted values. Spread of error e(i) must be same for all values of y(i). This property is called homoscedasticity. Therefore, heteroscedasticity should not exist in model.

In [199...

```
plt.scatter(x = y_hat, y = error)
plt.axhline(y=0)
plt.xlabel("predicted Value")
plt.ylabel("Residuals")
plt.title("Predicted values vs Residuals")
plt.show()
```



A little bit variation can be observed. This may conclude that some datapoints are outliers.

1. Remember three datapoints that are outlier, since the percentage of outliers was so low, we didn't remove them.
2. Errors are too low. They are nearly 0.

Actionable Insights & Recommendations

Insights from EDA of Dataset

1. The dataset has 500 records or entries with 9 features.
2. Datatype of features like Serial No., GRE Score, TOEFL Score, University Rating and Research is of integer type.
3. Datatype of features like SOP,LOR,CGPA and Chance of Admit is of float type.
4. There are no null values in dataset.
5. **Statistical Summary of all features of dataset has been commented along with code.**
6. Features like 'GRE Score', 'TOEFL Score' and 'CGPA' holds continuous numeric values.
7. Features like 'University Rating', 'SOP' and 'LOR' holds discrete numeric values.
8. 'University Rating' ranges from 1 to 5 whereas 'SOP' and 'LOR' features ranges from 0 to 8.
9. Feature 'Research' holds binary values either 0 or 1.
10. **Observations from univariate analysis have been commented along with code.**
11. There are outliers present only in 'LOR' and 'Chance of Admit' feature.
12. The percentage of outlier present in feature 'LOR' is 0.2% and 'Chance of Admit' is 0.4%.

Insights from DATA PREPROCESSING of Dataset

1. There are no duplicate data points present in dataset.
2. There are no missing values in dataset.
3. Feature 'LOR' and feature 'Chance of Admit' has outliers. No outliers present in other features.
4. No data has been deleted, as outliers are not affecting dataset much.
5. As values of all features are numerical and within certain range, there is no need of adding any new features.
6. Feature 'GRE Score' has maximum relation with features 'TOEFL score' and 'CGPA' followed by feature 'Chance of Admit' in comparison with other features.
7. Feature 'TOEFL score' has maximum correlation with features 'GRE Score' and 'CGPA' followed by feature 'Chance of Admit' in comparison with other features.
8. Feature 'CGPA' has maximum relation with 'Chance of Admit' followed by features 'GRE Score' and 'TOEFL Score'.
9. Feature 'Chance of Admit' has highest correlation with 'CGPA', followed by 'GRE Score' and 'TOEFL Score'.

Insights from DATA MODELING and Evaluation of Model Performance

1. The R-squared and Adjusted R-squared value of model from statsmodel are 0.822 and 0.819, respectively.
2. The R-squared and Adjusted R-squared value of model from sklearn.linear_model are 0.825 and 0.823, respectively.
3. The MSE, RMSE, and MAE of model from sklearn are 0.0091571, 0.0956902, and 0.0677363.
4. Features such as CGPA, GRE Score and TOEFL Score have coefficients more than 0.1, which means an increase in these scores lead to maximum chances of admission
5. On the other hand, Features such as University Rating and Research have coefficients are too low, that means, these features affect the chance of admission but not as much as features like CGPA, GRE and TOEFL Score do.
6. CGPA is the most important feature whereas SOP is least important.
7. ***The R2 score, MSE and MAE of model for train data and test data are almost equal.***
8. ***This concludes that the model build on train data is working fine for test data also.***
9. All errors(MSE,RMSE,MAE) of simple LR model and ridge model are almost same.
10. Since model is not overfitting, Results for Linear and Ridge are the same.
11. The R2 and adjusted R2 score of simple LR model and ridge model is almost equal.
12. R2_score and Adjusted_r2 are almost the same. Hence there are no unnecessary independent variables in the data.

Insights from Testing the assumptions of the linear regression model

1. Independent variables are linear with dependent(target) variable. All features having continuous values follow normal distribution.
2. Some features are highly correlated to each other. Multicollinearity present in model.

3. Since the plot of residuals vs predicted values is not creating a cone type shape. Hence, there is no homoscedasticity present in the data.
4. Mean of residuals is nearly zero in simple linear regression model as well as in ridge model also.
5. The distribution of errors is gaussian distribution.

Insight from model

1. CGPA is the most important feature in making the prediction for the Chance of Admit.
2. Top 3 correlated features with the Chance of Admit are CGPA, GRE Score and TOEFL Score.
3. Following are the final model results on the test data:
    A. Highest coeffient of Feature CGPA : 0.573
    B. Second highest coefficient of feature GRE Score: 0.177
    C. least coeffient of feature SOP: 0.009
    D. MSE: 0.009
    E. RMSE: 0.095
    F. MAE: 0.067
    G. R2_score: 0.825
    H. Adjusted_R2: 0.823

Recommendations

1. The model can be used to attract a large number of students and learners by providing them with an estimate of their chances of getting into good Institutes. This information can be used to market company's services to students who are most likely to be interested.
2. The model could be used to target marketing campaigns to specific regions or demographics.
3. The model could be used to track the performance of our coaching programs and make improvements.
4. The model could be used to identify new opportunities for growth in the education market.
5. As CGPA, GRE Scores and TOEFL Scores are important features, company can organized workshops on "How to increase or attain good CGPA"
6. Company can provide learning path or guidance path to those students who got less probability for chance of admission
7. Company can organize seminars to raise awareness about the importance of research capabilities for college admissions. This will help students improve their chances of getting admitted to a better university.
8. Company needs to create awareness and marketing campaigns to reach students early, even in their undergraduate years. This will help us build a strong brand and reputation, and it will also give students time to prepare for their future.
9. Success stories based on this model can be highlighted on dashboard.